1 Einleitung

Moderne unternehmensweite IT-Umgebungen unterliegen zahlreichen Herausforderungen von zunehmender Brisanz. Dies wird verursacht durch

- heterogene Systemlandschaften
- unterschiedliche HW-Plattformen
- unterschiedliche Netze und Netzzugriffe (TCP/IP, SNA, HTTP)
- Verflechtung der Anwendungen mit den Unternehmen

Dadurch entwickeln sich Problemfelder, sei es bei Fusionen, durch Kooperationen oder auch nur durch Rationalisierungsmaßnahmen. Die Unternehmen fordern flexible und skalierbare Anwendungen, gleichzeitig soll die Transaktionssicherheit für Prozesse und Daten gewährleistet bleiben, obwohl die Geschäftsprozesse immer komplexer werden. Die wachsende Globalisierung geht selbstverständlich davon aus, dass Anwendungen im 7x24-Stunden-Betrieb laufen und hochverfügbar sind, um beispielsweise Internetzugriffe auf bestehende Anwendungen über Zeitzonen hinweg zu ermöglichen.

Der klassische Anwendungsserver *open*UTM bietet eine Ablaufumgebung, die allen diesen Anforderungen moderner unternehmenskritischer Anwendungen gewachsen ist, denn *open*UTM verbindet alle Standards und Vorteile von Transaktionsmonitor-Middleware und Message Queuing Systemen:

- Konsistenz der Daten und der Verarbeitung
- Hohe Verfügbarkeit der Anwendungen (nicht nur der Hardware)
- Hohen Durchsatz auch bei großen Benutzerzahlen, d.h. höchste Skalierbarkeit
- Flexibilität bezüglich Änderungen und Anpassungen des IT-Systems

*open*UTM ist Teil des umfassenden Angebots von **openSEAS**. Im Rahmen des Produktangebots **openSEAS** (Open Suite for Enterprise Application Servers) nutzen innovative Produkte die ausgereifte Technologie von *open*UTM:

- Web*Transactions* ermöglicht u.a. zusammen mit *open*UTM moderne eBusiness-Applikationen zu erstellen. Bestehende UTM-Anwendungen können unverändert mit Web*Transactions* in das World Wide Web übernommen und in Portale integriert werden.
- Biz*Transactions* ist das openSEAS-Werkzeug zum Erzeugen wiederverwendbarer Business-Objekte. Die Business-Objekte werden aus bestehenden Anwendungen erzeugt (z.B. BS2000/OSD- und OS/390-Anwendungen, Transaktionsanwendungen mit *open*UTM und SAP R/3) und können je nach Bedarf als JavaBeans, .Net-Komponenten oder Web-Services verfügbar gemacht werden.
- BeanConnect ist eine Familie von Adaptern gemäß der Java Connector Architecture (JCA) von Sun und bietet den standardisierten Anschluss von UTM-Anwendungen an J2EE-Applikationsserver, insbesondere an ORACLE AS 10g. ORACLE AS 10g ist der Applikationsserver, der als Bestandteil von openSEAS als Entwicklungs- und Ablaufplattform aller Geschäftsanwendungen, Portale und Web-Seiten im Grid Computing-Maßstab zur Verfügung steht.

1.1 Konzept und Zielgruppen dieses Handbuchs

Dieses Handbuch richtet sich an Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen.

Es enthält alle Informationen, um ein UTM-Anwendungsprogramm in UNIX und Windows zu erzeugen und eine UTM-Anwendung einzusetzen.

Dieses Handbuch gibt Ihnen in den ersten Kapiteln einen Überblick darüber, wie Sie eine UTM-Anwendung binden und welche Dateien zum Betrieb einer Anwendung notwendig sind. Jeweils eigene Kapitel befassen sich mit dem Starten und Beenden einer UTM-Anwendung und mit dem Programmaustausch bei laufender Anwendung. Die Besonderheiten, die Sie beim Betrieb einer UTM-Datenbank-Anwendung beachten müssen, sind zentral in einem gleichlautenden Kapitel zusammengestellt.

Ausführlich wird darauf eingegangen, wie sich Terminal-Benutzer und andere Clients an eine UTM-Anwendung anmelden können.

Zusätzlich gibt es eigene Kapitel über die Tools, die Ihnen für den Betrieb und die Kontrolle einer UTM-Produktivanwendung zur Verfügung stehen.

Kenntnisse des Betriebssystems werden vorausgesetzt.

1.2 Wegweiser durch die Dokumentation zu *open*UTM

In diesem Abschnitt erhalten Sie einen Überblick über die Handbücher zu *open*UTM und zum Produktumfeld von *open*UTM.

1.2.1 *open*UTM-Dokumentation

Die *open*UTM-Dokumentation besteht aus Handbüchern, einer Online-Hilfe für den grafischen Administrationsarbeitsplatz *open*UTM WinAdmin sowie einer Freigabemitteilung für jede Plattform, auf der *open*UTM freigegeben wird.

Es gibt Handbücher, die für alle Plattformen gültig sind, sowie Handbücher, die jeweils für BS2000/OSD bzw. für UNIX, Linux und Windows gelten.

Sämtliche Handbücher sind als PDF-Datei im Internet verfügbar unter der Adresse
`http://manuals.fujitsu-siemens.com/servers/man/man_de/utm_man.htm`.
Die meisten Handbücher können außerdem auch als gebundene Papierexemplare bestellt werden. Zusätzlich gibt es eine CD-ROM, die die komplette *open*UTM-Dokumentation enthält.

Die folgenden Abschnitte geben einen aufgabenbezogenen Überblick über die Dokumentation zu *open*UTM V5.2. Eine vollständige Liste der Dokumentation zu *open*UTM finden Sie im Literaturverzeichnis auf Seite 253.

Einführung und Überblick

Das Handbuch **Konzepte und Funktionen** gibt einen zusammenhängenden Überblick über die wesentlichen Funktionen, Leistungen und Einsatzmöglichkeiten von *open*UTM. Es enthält alle Informationen, die Sie zum Planen des UTM-Einsatzes und zum Design einer UTM-Anwendung benötigen. Sie erfahren, was *open*UTM ist, wie man mit *open*UTM arbeitet und wie *open*UTM in die Betriebssysteme BS2000/OSD, UNIX, Linux und Windows eingebettet ist.

Programmieren

- Zum Erstellen von Server-Anwendungen über die KDCS-Schnittstelle benötigen Sie das Handbuch **Anwendungen programmieren mit KDCS für COBOL, C und C++**, in dem die KDCS-Schnittstelle in der für COBOL, C und C++ gültigen Form beschrieben ist. Diese Schnittstelle umfasst sowohl die Basisfunktionen des universellen Transaktionsmonitors als auch die Aufrufe für verteilte Verarbeitung. Es wird auch die Zusammenarbeit mit Datenbanken beschrieben.

- Wollen Sie die X/Open-Schnittstellen nutzen, benötigen Sie das Handbuch **Anwendungen erstellen mit X/Open-Schnittstellen**. Es enthält die UTM-spezifischen Ergänzungen zu den X/Open-Programmschnittstellen TX, CPI-C und XATMI sowie Hinweise zu Konfiguration und Betrieb von UTM-Anwendungen, die X/Open-Schnittstellen nutzen. Ergänzend dazu benötigen Sie die X/Open-CAE-Specification für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument ***open*UTM Data Marshalling mit XML**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden. Diese Beschreibung ist nur als PDF-Dokument erhältlich (online und auf CD).
- Für BS2000/OSD gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1. Diese sind nur als PDF-Datei verfügbar (online und auf CD).

Konfigurieren

Zur Definition von Konfigurationen steht Ihnen das Handbuch **Anwendungen generieren** zur Verfügung. Darin ist beschrieben, wie Sie für eine UTM-Anwendung mit Hilfe des UTM-Tools KDCDEF die Konfiguration definieren und die KDCFILE erzeugen. Zusätzlich wird gezeigt, wie Sie wichtige Verwaltungs- und Benutzerdaten mit Hilfe des Tools KDCUPD in eine neue KDCFILE übertragen, z.B. beim Umstieg auf eine neue *open*UTM-Version oder nach Änderungen in der Konfiguration.

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000/OSD bzw. UNIX/Windows) das Handbuch **Einsatz von *open*UTM-Anwendungen**. Dort ist beschrieben, wie man ein UTM-Anwendungsprogramm bindet und startet, wie man sich bei einer UTM-Anwendung an- und abmeldet und wie man Anwendungsprogramme strukturiert und im laufenden Betrieb austauscht. Außerdem enthält es die UTM-Kommandos, die dem Terminal-Benutzer zur Verfügung stehen.

Administrieren und Konfiguration dynamisch ändern

- Für das Administrieren von Anwendungen finden Sie die Beschreibung der Programmschnittstelle zur Administration und die UTM-Administrationskommandos im Handbuch **Anwendungen administrieren**. Es informiert über die Erstellung eigener Administrationsprogramme und über die Möglichkeiten der zentralen Administration mehrerer Anwendungen. Darüber hinaus beschreibt es, wie Sie Message Queues und Drucker mit Hilfe der KDCS-Aufrufe DADM und PADM administrieren können.

- Wenn Sie den grafischen Administrationsarbeitsplatz *open***UTM WinAdmin** einsetzen, dann steht Ihnen folgende Dokumentation zur Verfügung:
  - Die **WinAdmin-Beschreibung** bietet einen umfassenden Überblick über den Funktionsumfang von WinAdmin und das Handling von WinAdmin. Dieses Dokument wird mit der Software ausgeliefert und ist zusätzlich auch online als PDF-Datei verfügbar.
  - Das **Online-Hilfesystem** beschreibt kontextsensitiv alle Dialogfelder und die zugehörigen Parameter, die die grafische Oberfläche bietet. Außerdem wird dargestellt, wie man WinAdmin konfiguriert, um die UTM-Anwendungen administrieren und generieren zu können.

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für die Betriebssysteme UNIX/Windows und für das Betriebssystem BS2000/OSD). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das Verhalten im Fehlerfall, das Meldungswesen von *open*UTM, sowie alle von *open*UTM ausgegebenen Meldungen und Returncodes.

*open*UTM-Clients erstellen

Wenn Sie Client-Anwendungen für die Kommunikation mit UTM-Anwendungen erstellen wollen, stehen Ihnen folgende Handbücher zur Verfügung:

- Das Handbuch ***open*UTM-Client für Trägersystem UPIC** beschreibt Erstellung und Einsatz von Client-Anwendungen, die auf UPIC basieren. Neben der Beschreibung der Schnittstellen CPI-C und XATMI erhalten Sie Informationen, wie Sie die C++-Klassen oder ActiveX für die schnelle und einfache Programmerstellung nutzen können.
- Das Handbuch ***open*UTM-Client für Trägersystem OpenCPIC** beschreibt, wie man OpenCPIC installiert und konfiguriert. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.
- Für die mit BeanConnect ausgelieferten **JUpic-Java-Klassen** wird die Dokumentation mit der Software ausgeliefert. Diese Dokumentation besteht aus Word- und PDF-Dateien, die die Einführung und die Installation beschreiben, sowie aus einer Java-Dokumentation mit der Beschreibung der Java-Klassen.

Kopplung mit der IBM-Welt

Wenn Sie aus Ihrer UTM-Anwendung mit Transaktionssystemen von IBM kommunizieren wollen, benötigen Sie außerdem das Handbuch **Verteilte Transaktionsverarbeitung zwischen *open*UTM und CICS-, IMS- und LU6.2-Anwendungen**. Es beschreibt die CICS-Kommandos, IMS-Makros und UTM-Aufrufe, die für die Kopplung von UTM-Anwendungen mit CICS- und IMS-Anwendungen benötigt werden. Die Kopplungsmöglichkeiten werden anhand ausführlicher Konfigurations- und Generierungsbeispiele erläutert. Außerdem beschreibt es die Kommunikation über *open*UTM-LU62, sowie dessen Installation, Generierung und Administration.

1.2.2 Dokumentation zum openSEAS-Produktumfeld

Die Verbindung von *open*UTM zum openSEAS-Produktumfeld wird im *open*UTM-Handbuch **Konzepte und Funktionen** kurz dargestellt. Die folgenden Abschnitte zeigen, welche der openSEAS-Dokumentationen für *open*UTM von Bedeutung sind.

Web-Anbindung

Zum Anschließen neuer und bestehender UTM-Anwendungen an das Web mit dem Produkt Web*Transactions* benötigen Sie die folgenden Handbücher:

- Das einführende Handbuch **Web*Transactions* - Konzepte und Funktionen** gibt Ihnen einen Überblick über die Leistungsfähigkeit und Einsatzmöglichkeiten des Produkts Web*Transactions* und informiert Sie über dessen Eigenschaften und Funktionsweisen. Das Handbuch erläutert das Objekt-Konzept und den dynamischen Ablauf einer Web*Transactions*-Sitzung.
- Im Handbuch **Web*Transactions* - Template-Sprache** sind alle Sprachmittel der Template-Sprache WTML erläutert. Es enthält zahlreiche Beispiele, die die Sprachmittel veranschaulichen und Ihnen den Einsatz erleichtern.
- Die Handbücher **Web*Transactions* - Anschluss an *open*UTM-Anwendungen über UPIC** und **Web*Transactions* - Anschluss an OSD-Anwendungen** beschreiben die notwendigen Schritte, um UTM-Dialoganwendungen über die UPIC- bzw. die Terminal-Schnittstelle an das Web anzuschließen. An einem konkreten Beispiel werden diese Schritte nochmals verdeutlicht.

Anwendungsintegration

Wenn Sie UTM-Anwendungen und -Services in Microsoft-Anwendungen integrieren möchten, dann steht Ihnen dafür das Produkt Biz*Transactions* mit dem Handbuch **Biz*Transactions* - Anwendungsintegration mit Business Objekten** zur Verfügung. Es beschreibt neben der Installation und Konfiguration auch die Konzepte und Komponenten von Biz*Transactions*. Außerdem wird die Administration und das Entwickeln von integrierten Client-Anwendungen erläutert.
Für Biz*Transactions* wird mit der Software auch ein Online-Hilfesystem ausgeliefert.

Ablaufumgebung für Enterprise JavaBeans

Die Dokumentation zu BeanConnect ist in Vorbereitung.

Die Handbücher zu ORACLE AS 10g sind bei ORACLE beziehbar.

1.2.3 Readme-Dateien

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. den produktspezifischen Readme-Dateien.

- UNIX:

  Letzte Änderungen zum vorliegenden Handbuch können in einer so genannten man-Datei (Handbuchergänzungsdatei) hinterlegt sein. In diesem Fall enthält die Lieferinformation einen Hinweis darauf. Die Freigabemitteilungen stehen bei allen UNIX-Systemen im Installationsverzeichnis unter `/docs/`*`sprache`*.

  Readme- und man-Dateien sind ASCII-Dateien, die Sie mit einem Editor ansehen oder auf einem Standarddrucker ausdrucken können.

- Windows:

  Die Readme-Datei und ggf. weitere Dateien wie z.B. eine Handbuchergänzungsdatei finden Sie im UTM-Installationsverzeichnis unter `\Docs\`*`sprache`*.

1.3 Neuerungen in *open*UTM V5.2

*open*UTM bietet eine Reihe neuer Funktionen auf Server- und auf Client-Seite. Der grafische Administrationsarbeitsplatz WinAdmin wurde ebenfalls funktionell erweitert.

Die folgenden Abschnitte gehen näher auf die Neuerungen in den einzelnen Bereichen ein.

1.3.1 Neue Plattformen

- *open*UTM bietet auf Solaris-Systemen volle Ablauffähigkeit im **64-bit-Modus**. Der Ablaufmodus (32/64-bit) kann über die Administrationsschnittstelle KDCADMI abgefragt werden.
- Der *open*UTM-Server und *open*UTM-Client laufen auch auf den aktuellen Windows-Plattformen Windows XP und Windows 2003.

1.3.2 Neue Server-Funktionen

Erweiterungen beim Message Queuing

- Browse-Funktion

  Mit der neuen Browse-Funktion können Nachrichten von Service-gesteuerten Queues gelesen werden, ohne sie zu löschen. Damit lassen sich Nachrichten einer Queue auch parallel von mehreren Vorgängen lesen.

  Zusammen mit der Browse-Funktion ist es möglich, Nachrichten einer Service-gesteuerten Queue gezielt zu verarbeiten (nicht nur sequenziell).

- Redelivery-Funktion

  Für Nachrichten in Service-gesteuerten Queues und Nachrichten an Hintergrundaufträge ist eine erneute Zustellung möglich, wenn während der Verarbeitung ein Fehler auftritt (abnormales Beenden des Asynchron-Services bzw. Rücksetzen der Transaktion). Die Anzahl der erneuten Zustellungen lässt sich im Programm ermitteln, die maximale Anzahl ist per Generierung einstellbar.

Verschlüsselung nach dem AES/RSA-Verfahren

Die Verschlüsselung ist wahlweise nach dem DES/RSA oder dem AES/RSA-Verfahren möglich. Für den RSA-Schlüssel sind die Modulo-Längen 200, 512, 1024 und 2048 möglich. Für den Zugangsschutz gegenüber Clients sind damit 4 Verschlüsselungsebenen möglich, abhängig von der Länge des RSA-Schlüssels. Beim Zugriffsschutz können 2 Ebenen definiert werden (DES- oder AES-Verschlüsselung).

Alle RSA-Schlüssel können per programmierter Administration (KDCADMI) neu erzeugt, gelesen, aktiviert, deaktiviert oder gelöscht werden.

Erweiterung beim XML Data Marshalling

- Es können Namensräume definiert werden, um in einem XML-Dokument mehrere Namensspezifikationen zu verwenden.
- Es lassen sich mehrere Kindknoten mit gleichem Namen erzeugen. Auf diese kann mit Lese- und Schreibaufrufen zugegriffen werden.
- Daten können umcodiert und Zeichensätze können ermittelt/gesetzt werden.
- Attribute können erzeugt werden, auf die mit Lese-/Schreib- und Positionieraufrufen zugegriffen werden kann.

Sonstige Änderungen

- Erweiterte Transaktionssteuerung:
  Eine Transaktion kann jetzt so beendet oder zurückgesetzt werden, dass der Programmkontext erhalten bleibt und das Programm anschließend fortgesetzt werden kann (neue PGWT-Varianten).
- Die Namen von Teilprogrammen können zusätzliche Sonderzeichen enthalten.
- In BS2000/OSD werden UTM-Systemcode, KDCROOT-Modul und Dienstprogramme über LLMs geladen. Damit sind Korrekturen für einzelne Komponenten mittels Software-Korrekturverfahren möglich (z.B. Austausch ganzer Module).
- In BS2000/OSD ist bei KDCUPD die Übertragung der „Locales“ steuerbar.
- In BS2000/OSD-Teilprogrammen können die Druckoptionen für RSO-Drucker übergeben werden.
- Für BS2000/OSD-Datenbanken kann das Datenbank-Verbindungsmodul aus dem IMON-Installationspfad nachgeladen werden. Dadurch ist die UTM-Anwendung unabhängig von Installationspfaden und Bibliotheksnamen des Datenbanksystems.

1.3.3 Neue Client-Funktionen

- Erweiterung der Sicherheitsfunktionen
  - Der UPIC-Client unterstützt alle Verschlüsselungsebenen des *open*UTM-Servers. Verschlüsselung nach dem DES/RSA bzw. AES/RSA-Verfahren mit Modulo-Länge des Schlüssels von 200, 512, 1024 und 2048.
  - Der UPIC-Client kann das detaillierte Ergebnis der Berechtigungsprüfung in Form eines erweiterten Returncodes ermitteln.
- Neue Funktionen zum Setzen von Conversation Characteristics

  Die Conversation Characteristics CHARACTER_CONVERTION (ASCII-EBCDIC-Konvertierung) und COMMUNICATION_PROTOCOL sind im Programm veränderbar.
- Verbesserte Steuerung des Wiederanlaufs

  Der Client kann einen 8 byte langen Kontext an *open*UTM übergeben und nach einem Wiederanlauf wieder auslesen.
- Abfragen des Server-Status

  Der Client kann den aktuellen Vorgangs- und Transaktionsstatus des Servers abfragen.
- Weiterentwicklung des ActiveX Control

  Das Control wurde um die Properties *PartnerLUName* und *PartnerPort* erweitert.

1.3.4 Erweiterungen für *open*UTM WinAdmin

- Es können benutzerspezifische Aktionen definiert werden wie z.B. das Öffnen von Dateien, Speichern von Listen in Dateien und Aufrufen von Skripts und Programmen.
- Die Statistik-Funktionen wurden erweitert um Schwellwert-Alarm, Erfassung von Statistikwerten einzelner UTM-Objekte, verbesserte Statistik-Grafik und Ausdrucken der Statistik-Grafik.
- Die Bean*Transactions* Management Console kann von WinAdmin aus aufgerufen werden und der Import von Bean*Transactions*-Dateien ist möglich (Container-Properties).
- Die erweiterten Encryption-Funktionen von *open*UTM V5.2 können per WinAdmin administriert werden. Mit WinAdmin kann man RSA-Schlüssel für unterschiedliche Verschlüsselungsebenen erzeugen, lesen, aktivieren/deaktivieren und löschen.
- Anzeige des OSI-LPAP in der Liste der generierten OSI-Associations.

1.4 Darstellungsmittel

Metasyntax

Die in diesem Handbuch verwendete Metasyntax können Sie der folgenden Tabelle entnehmen:

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Konstanten (Namen von Aufrufen, Anweisungen, Feldnamen, Kommandos und Operanden etc.), die in dieser Form anzugeben sind.	`LOAD-MODE=STARTUP`
kleinbuchstaben	In Kleinbuchstaben sind in Syntaxdiagrammen und Operandenbeschreibung die Platzhalter für Operandenwerte dargestellt.	`KDCFILE=filebase`
kleinbuchstaben	Im Fließtext werden Variablen sowie Namen von Datenstrukturen und Feldern in kursiven Kleinbuchstaben dargestellt.	*utmpfad* ist das UTM-Installationsverzeichnis
`Schreibmaschinen-schrift`	In Schreibmaschinenschrift werden im Fließtext Kommandos, Dateinamen, Meldungen und Beispiele ausgezeichnet, die in genau dieser Form eingegeben werden müssen bzw. die genau diesen Namen oder diese Form besitzen.	Der Aufruf `tpcall`
{ } und |	In geschweiften Klammern stehen alternative Angaben, von denen Sie eine auswählen müssen. Die zur Verfügung stehenden Alternativen werden jeweils durch einen Strich getrennt aufgelistet.	`STATUS={ ON | OFF }`
[ ]	In eckigen Klammern stehen wahlfreie Angaben, die entfallen können.	`KDCFILE=( filebase [, { SINGLE| DOUBLE} ] )`
( )	Kann für einen Operanden eine Liste von Parametern angegeben werden, sind diese in runde Klammern einzuschließen und durch Kommata zu trennen. Wird nur ein Parameter angegeben, kann auf die Klammern verzichtet werden.	`KEYS=(key1,key2,...key`*n*`)`
<u>Unterstreichen</u>	Unterstreichen kennzeichnet den Standardwert.	`CONNECT= { A/YES | NO }`

Formale Darstellung	Erläuterung	Beispiel
Kurzform	Die Standardkurzform für Anweisungen, Operanden und Operandenwerte wird „fett“ hervorgehoben. Die Kurzform kann alternativ angegeben werden.	**T**RANSPORT-**SEL**EKTOR=c‘C‘
...	Punkte zeigen die Wiederholbarkeit einer syntaktischen Einheit an. Außerdem kennzeichnen die Punkte Ausschnitte aus einem Programm, einer Syntaxbeschreibung o.ä.	KDCDEF starten : : OPTION DATA=statement_file : END

Sonstige Symbole

Alle Stellen, die mit dem nebenstehenden Symbol gekennzeichnet sind, verweisen auf umfassende und detaillierte Informationen zum jeweiligen Thema.

für Hinweistexte

für Warnhinweise

2 Anwendungsprogramm erzeugen

Ein UTM-Anwendungsprogramm besteht aus einer Reihe von Modulen, die spätestens zur Ablaufzeit als ein Programm gebunden sein müssen.

Die Teilprogramme definieren die Anwendungslogik und müssen vor dem Start der Anwendung geschrieben und übersetzt werden. Sehen Sie dazu das *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“.

Damit die Teilprogramme unter *open*UTM ablaufen können, wird das UTM-Anwendungsprogramm wie folgt erzeugt:

- ROOT-Tabellen-Source, die KDCDEF erzeugt hat, übersetzen.
- ROOT-Tabellen, UTM-Main-Routine, UTM-Systemmodulen für die Main Routine KDCROOT, C-Laufzeitsystem und eventuell weitere Laufzeitsysteme, Meldungsmodul, Benutzerbibliotheken und Teilprogramme binden.

Die Teilprogramme können Sie auch als Shared Objects binden. Shared Objects können im laufenden Betrieb dynamisch ausgetauscht werden.

Informationen zum Programmaustausch mit Shared Objects finden Sie Kapitel „Programmaustausch im Betrieb“ auf Seite 119.

Die einzelnen Schritte, die dabei zum Erzeugen eines UTM-Anwendungsprogramms notwendig sind, zeigt die folgende Abbildung.

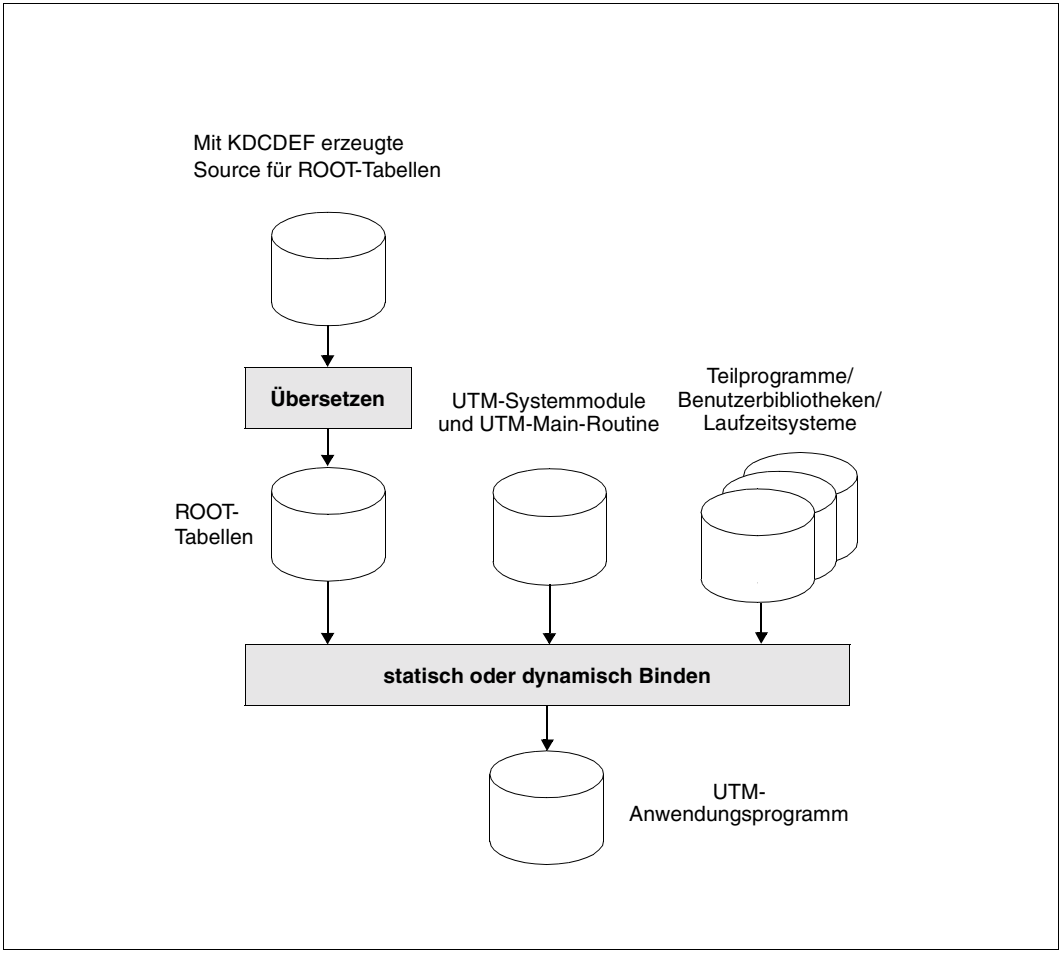


Bild 1: Übersicht: Erzeugen des UTM-Anwendungsprogramms

Main Routine KDCROOT

Aus den ROOT-Tabellen, der UTM-Main-Routine, die bei der Installation von *open*UTM erzeugt wird, und den UTM-Systemfunktionen, entsteht beim Binden die Main Routine KDCROOT als Teil des Anwendungsprogramms. KDCROOT fungiert beim Ablauf der Anwendung als steuerndes Hauptprogramm und übernimmt u. a. folgende Aufgaben:

– Verbindung zwischen den Teilprogrammen und den UTM-Systemfunktionen
– Ablauf-Koordination von Teilprogrammen unterschiedlicher Programmiersprachen
– Kopplung zu Datenbanken

Weiter enthält KDCROOT Bereiche für variable Daten sowie die Nachrichtenbereiche. Nähere Informationen zur Main Routine KDCROOT sind dem *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“ zu entnehmen.

KDCDEF erzeugt das ROOT-Tabellenmodul als C/C++-Source, die Sie mit dem C/C++-Compiler übersetzen und mit Ihren Teilprogrammen, den UTM-Systemmodulen sowie eventuell weiteren Modulen zu einem ablauffähigen Programm binden, siehe nächster Abschnitt.

2.1 UTM-Prozess binden unter UNIX

Für eine UTM-Anwendung müssen Sie die unten aufgelisteten Module in der angegebenen Reihenfolge zu einem Workprozess binden, damit alle Externverweise aufgelöst werden können. Den Workprozess müssen Sie im Dateiverzeichnis *filebase* Ihrer Anwendung unter dem Namen *utmwork* ablegen.

Wenn Sie die Funktion „Programmaustausch“ nutzen wollen, gibt es zwei Möglichkeiten:

- Wenn Sie Teile des Anwendungsprogramms austauschen möchten, müssen Sie Shared Objects verwenden. Diese müssen Sie mit den für Shared Objects notwendigen entsprechenden Optionen übersetzen und die Anwendung dynamisch binden.
- Wenn nur der komplette Workprozess austauschbar sein soll, müssen Sie beim Binden nichts beachten. Nach dem Binden müssen Sie *utmwork* mit Hilfe des Tools KDCPROG in das Dateigenerationsverzeichnis *filebase*/PROG bringen.

Sie können sich die Arbeit erleichtern, wenn Sie mit Hilfe der ausgelieferten Beispielanwendung ein Makefile erzeugen, das Sie als Grundlage für Ihre Bindeprozedur verwenden, siehe „Binden mit Makefile“ auf Seite 20.

Wenn Sie COBOL-Teilprogramme verwenden, dann müssen Sie die Umgebungsvariable COBDIR für den Cobol-Compiler entsprechend der COBOL-Dokumentation setzen, typischerweise auf `/opt/lib/cobol`. Ebenso ist die Umgebungsvariable LD_LIBRARY_PATH auf `/opt/lib/cobol/lib` zu setzen.

2.1.1 Benötigte Objektmodule

Die im Folgenden aufgelisteten Objektmodule können beim Binden einer Anwendung angegeben werden. Welche dieser Module zum Ablauf notwendig sind (d.h. mit eingebunden werden müssen), ist abhängig vom Umfang der jeweiligen UTM-Anwendung.

- Die Main Routine von *open*UTM

  Hier müssen Sie entweder das Objekt `mainutm.o` einbinden, falls nur C- und COBOL-Programme verwendet werden oder das Objekt `mainutmCC.o`, falls mindestens ein C++-Programm verwendet wird.

  Auf den meisten Systemen darf die Anwendung nicht gleichzeitig C++- und COBOL-Programme enthalten.

  Die Objekte `mainutm.o` und `mainutmCC.o` stehen in *utm-dateiverzeichnis*`/sys`. Das Objekt `mainutmCC.o` wird während der Installation von *open*UTM erzeugt, wenn der C++-Compiler bereits installiert ist (d.h. wenn das Kommando `type CC` zum Erfolg führt).

Wird der C++-Compiler erst nach *open*UTM installiert oder konnte *open*UTM den C++-Compiler bei der Installation aus irgendwelchen anderen Gründen nicht finden, dann können Sie `mainutmCC.o` mit Hilfe des UTM-Shellskripts `CCmainutm` selbst erzeugen. Dazu müssen Sie folgendes Kommando eingeben:

`UTMPATH=`*utm-dateiverzeichnis* `\`
utm-dateiverzeichnis`/shsc/CCmainutm`

Näheres zum *utm-dateiverzeichnis* siehe Seite 186.

- Die Main Routine KDCROOT (`rootname.o`)

  Dieses Modul wird in jeder Anwendung benötigt und aus dem ROOT-Quellprogramm erzeugt. Das ROOT-Quellprogramm (*filebase*`/`*rootname*`.c`) wird beim KDCDEF-Lauf erzeugt , wenn Sie in der OPTION-Anweisung GEN=ROOTSRC oder GEN=ALL angeben. *rootname* definieren Sie in der ROOT-Anweisung und *filebase* in der MAX-Anweisung mit dem Parameter KDCFILE=.

  Sie erzeugen KDCROOT wie folgt:

  – Übersetzen Sie das mit KDCDEF erstellte ROOT-Quellprogramm mit dem C-Compiler. Wenn Sie COBOL-Teilprogramme verwenden, müssen Sie zusätzlich den Schalter `-I$COBDIR/include` angeben.

  – Speichern Sie Ausgabe unter *filebase/rootname*`.o`.

  Sie müssen *filebase/rootname*`.o` beim Binden der Anwendung **vor** den anwendungsspezifischen Modulen angeben.

- Ihre übersetzten Teilprogramme

  Ihre übersetzten Teilprogramme können Sie entweder als einzelne Objekte oder als Teile von Objektbibliotheken einbinden. Dabei kann es sich um statische Bibliotheken oder um dynamische Bibliotheken (genannt Shared Objects) handeln. Wenn Sie Teilprogramme als Shared Objects einsetzen, dann müssen diese so übersetzt werden, dass dabei das jeweilige Laufzeitsystem (C, C++ oder COBOL) mit eingebunden wird.

  Für C++-Programme ist auf einigen Plattformen wie z.B. PrimePower Solaris folgender Aufruf notwendig, um die Laufzeitbibliothek *libCstd.a* explizit einzubinden:

  `CC -G -o` *shared-object .. objects...* `-lCstd`

  Wenn Sie COBOL-Teilprogramme verwenden, dann beachten Sie bitte:

  – *open*UTM V5.2 setzt den COBOL-Compiler Server Express ab V2.2 von MicroFocus voraus.

– In den neuen COBOL-Compilern gibt es die Schlüsselwörter OBJECT-ID und RESTRICTED. Diese Schlüsselwörter kollidieren mit Namen in den COPY-Elementen der UTM-Schnittstellen. Um diese Konflikte zu vermeiden, haben Sie zwei Möglichkeiten:

  a) Wenn die COBOL-Teilprogramme nicht tatsächlich objektorientiert sind, dann geben Sie beim Übersetzen die Compiler-Schalter REMOVE(OBJECT-ID) und REMOVE(RESTRICTED) an.

  b) Wenn die objektorientierte Funktionalität erhalten bleiben soll, dann müssen die COPY-Anweisungen wie folgt modifiziert werden:

     `COPY` *COPY-Element* `REPLACING OBJECT-ID BY` *NEW-OBJECT-ID.*

     `COPY` *COPY-Element* `REPLACING RESTRICTED BY` *NEW-RESTRICTED.*

     Beim Datenzugriff müssen die neuen Namen verwendet werden, siehe auch *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“.

– Shared Objects erzeugen Sie mit folgendem Aufruf:

  `cob -z -o` *shared-object Cobol-objects*

  Die Reihenfolge der Schalter und der angegebenen Objekte muss unbedingt eingehalten werden.

– Wenn Sie 64-bit-fähige Anwendungen erzeugen möchten, dann müssen Sie *cob64* statt *cob* verwenden. Falls *cob64* auf *cob* gelinkt ist, dann erzeugt *cob* schon eine 64-bit-Anwendung. In diesem Fall muss eine 32-bit-Anwendung mit *cob32* statt *cob* übersetzt werden.

- Die UTM-Systemmodule einschließlich Administrationsprogramm

  Diese Module werden in jedem Anwendungsprogramm benötigt. Die Module für das statische Binden stehen in der Bibliothek *utm-dateiverzeichnis*`/sys/libwork.a` zur Verfügung. Auf Anlagen, die dynamisches Binden unterstützen, steht für das dynamische Binden die Bibliothek *utm-dateiverzeichnis*`/sys/libwork.so` zur Verfügung.

  Zusätzlich muss die Bibliothek *utm-dateiverzeichnis*`/sys/libutmcrypt.a` (statisch) bzw. *utm-dateiverzeichnis*`/sys/libutmcrypt.so` (dynamisch) mit eingebunden werden.

  Wenn Sie mit SHARED-OBJECT generierte Shared Objects in Ihrer Anwendung einsetzen, müssen Sie immer dynamisch binden.

- OSI-TP-Module (optional)

  Falls Kommunikation über OSI TP betrieben werden soll, muss entweder die Bibliothek *utm-dateiverzeichnis*`/sys/libxaptp.a` (statisch) oder die Bibliothek *utm-dateiverzeichnis*`/sys/libxaptp.so` (dynamisch) eingebunden werden.

  Zusätzlich werden in beiden Fällen die OSS-Bibliotheken `libossutm.so` und `liboss.so` benötigt. Dabei müssen Sie `libossutm.so` immer **vor** `liboss.so` angeben, sonst kommt es zu folgendem Fehler beim Start der Anwendung:

```
P001 Fehler beim OSS Aufruf (o_create() ·): – 1, 300, 199, 0
K060 Der Anwendungslauf wurde abgebrochen; die Ursache ist XINI06.
```

- Die Laufzeitsysteme für C und evtl. COBOL

  Das Laufzeitsystem für die Programmiersprache C wird immer benötigt. Das COBOL-Laufzeitsystem wird nur benötigt, wenn Sie ein Teilprogramm in COBOL erstellt haben.

- Ein benutzereigenes Meldungsmodul (optional)

  Wenn Sie ein eigenes Meldungsmodul erstellt haben (siehe *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“), müssen Sie dieses Objektmodul zusätzlich einbinden.

- Weitere Laufzeitsysteme für Datenbank- bzw. Formatierungssystem (optional)

  Welche Bibliotheken und Module Sie für eine UTM-Datenbankkopplung einbinden müssen, ist in Abschnitt „UTM-Datenbank-Anwendung binden in UNIX“ auf Seite 74f beschrieben.

- Weitere Bibliotheken für XML (optional)

  Falls Sie in Teilprogrammen die XML-Funktionen nutzen möchten, benötigen Sie die entsprechende XML-Bibliothek.

  Näheres siehe Dokumentation zur XML-Schnittstelle.

- Weitere Bibliotheken für X/Open (optional)

  Falls Sie Teilprogramme mit den X/Open-Schnittstellen erstellt haben, benötigen Sie die UTM-Bibliothek *utm-dateiverzeichnis*`/sys/libxopen.a` (statisch) bzw. *utm-dateiverzeichnis*`/sys/libxopen.so` (dynamisch). Zusätzlich muss beim Binden die Option `-lm` angegeben werden.

- Auf Linux müssen Sie die Option `-lcrypt` angeben.

- Die Option `-ldl` müssen Sie immer angeben.

2.1.2 Binder aufrufen

Je nachdem, in welcher Sprache die Teilprogramme erstellt wurden, müssen Sie folgenden Binder verwenden:

– den C-Binder `cc`, falls nur C-Programme enthalten sind,

– den COBOL-Binder, falls mindestens ein COBOL-Programm enthalten ist. Der Aufruf lautet dann:

`cob -o utmwork` *shared-object UTM-Systembibliotheken*

Für 64-bit-Anwendungen müssen Sie *cob64* statt *cob* verwenden.

– den C++-Binder CC, falls mindestens ein C++-Programm enthalten ist. COBOL-Programme dürfen dann nicht vorhanden sein.

2.1.3 Tätigkeiten nach dem Binden

Das gebundene Anwendungsprogramm muss im Verzeichnis *filebase* unter dem Namen `utmwork` abgelegt werden. Nach jedem Binden einer Anwendung muss mit der Prozedur

utm-dateiverzeichnis`/shsc/nmwork`

eine Datei mit der zugehörigen Symboltabelle neu erzeugt werden. Diese Prozedur benötigt das Dateiverzeichnis *filebase* als Parameter.

2.1.4 Binden mit Makefile

Für Ihr Anwendungsprogramm können Sie ein Makefile erstellen, indem Sie das Makefile der Beispielanwendung, die zusammen mit *open*UTM ausgeliefert wird, anwendungsspezifisch abändern.

Die Beispielanwendung finden Sie nach der Installation im CPIO-Archiv *utm-dateiverzeichnis*`/CPIO.utmsample`. Sie können die Beispielanwendung mit der Prozedur *utm-dateiverzeichnis*`/shsc/install.sample` in der Benutzerkennung installieren, unter der auch Ihre UTM-Anwendung ablaufen soll. Erstellen Sie mit der Prozedur `p/config` eine Konfiguration der Beispielanwendung, die Ihrer Anwendung entspricht. Das Makefile `makefile` dieser Beispielanwendung können Sie dann für Ihre Anwendung ändern und erweitern, siehe Online-Dokumentation der Beispielanwendung.

Wenn Sie mit einer Datenbank koppeln möchten, geben Sie beim `p/config`-Lauf das verwendete Datenbanksystem und die benötigte Konfiguration an. Das von `p/config` erzeugte Makefile enthält dann i.a. alle benötigten Module und Bibliotheken für die Datenbankkopp-

lung. Die Bibliothekslisten werden unter `.liblists/INFORMIX`, `/INFOMMIXXA`, `/INFORMIXCOB` bzw. `/ORACLE` erzeugt. Näheres zum Binden einer UTM-Datenbank-Anwendung finden Sie in Kapitel „UTM-Datenbank-Anwendung“ auf Seite 73.

2.2 Anwendungsprogramm erzeugen unter Windows

Unter Windows können Sie Anwendungsprogramme in C, C++ oder in COBOL erstellen.

Anwendungsprogramme in C und C++

Für Anwendungsprogramme in C und C++ müssen Sie mit dem Microsoft Visual C++ Developer-Studio ab V5.0 arbeiten. Dazu sind folgende Schritte nötig:

1. Optionen des Developer Studios einstellen
2. Projekt erzeugen für die Anwendung
3. Programme schreiben oder vorhandene Programme modifizieren
4. Anwendung übersetzen und Binden
5. Anwendung als Dienst installieren, falls gewünscht.
   Dies hat u.a. den Vorteil, dass die Anwendung automatisch mit dem System gestartet und beendet werden kann.

Die Abschnitte „Optionen des Developer Studios einstellen“ auf Seite 22 bis „Anwendung übersetzen und binden“ auf Seite 26 beschreiben, wie Sie ein statisch gebundenes Anwendungsprogramm erzeugen.

Wenn Sie Anwendungsprogramme dynamisch nachladen wollen, müssen Sie diese als DLLs erstellen, siehe Abschnitt „Anwendungsprogramme als DLLs erstellen“ auf Seite 30.

Was Sie für Anwendungsprogramm in COBOL beachten müssen, ist im Abschnitt „COBOL-Anwendungsprogramme in Windows“ auf Seite 30 beschrieben.

Die folgende Beschreibung gilt für das Developer Studio V5.0.

Das Vorgehen mit Microsoft Developer Studio V6.0 bzw. Microsoft Visual Studio .NET 2003 ist entsprechend. Bitte beachten Sie mögliche Hinweise in der Manual-Ergänzungsdatei oder der Freigabemitteilung.

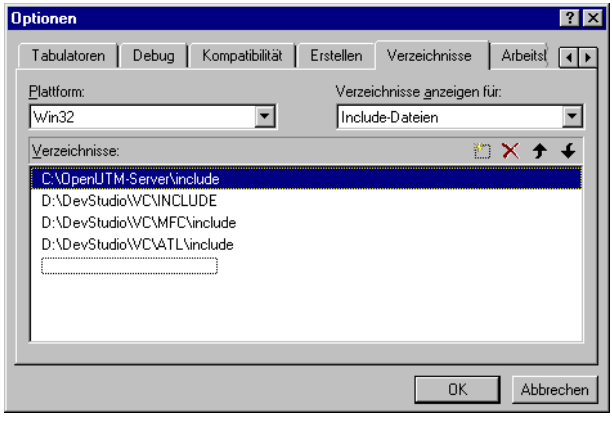
2.2.1 Optionen des Developer Studios einstellen

Bevor Sie UTM-Anwendungen zu entwickeln beginnen, müssen Sie die Verzeichnisse mit den *open*UTM Include Files (*utm-dateiverzeichnis*`\include`), den *open*UTM Library Files (*utm-dateiverzeichnis*`\sys`) und ggf. den Datenbank-Bibliotheken in die Entwicklungsumgebung aufnehmen.

Dies stellen Sie über bestimmte Optionen des Developer Studios ein. Diese Einstellungen sind projektunabhängig und gelten daher für die Erstellung unterschiedlicher UTM-Anwendungen.

Gehen Sie wie folgt vor:

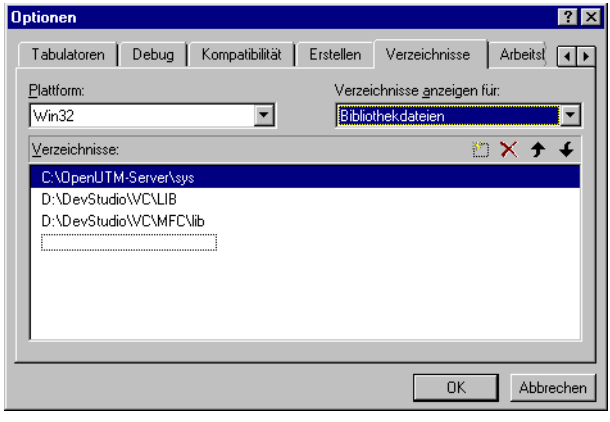
1. Rufen Sie das Microsoft Visual C++ Developer Studio auf.
2. Wählen Sie *Extras - Optionen* und klicken Sie das Blatt *Verzeichnisse* an.
3. Setzen Sie wie folgt die Option für die Include-Dateien:
   - Wählen Sie in der Box *Verzeichnisse anzeigen für* den Wert *Include-Dateien* aus und doppelklicken Sie in die leere Eingabezeile.
   - Klicken Sie rechts auf „ ... “.
     Damit erscheint das Dialogfeld *Verzeichnis wählen*. Navigieren Sie dort in das Installationsverzeichnis von *open*UTM (Standard ist `C:\OpenUTM-Server`) und wählen Sie darin das Verzeichnis `include` aus.
   - Klicken Sie anschließend OK; Sie erhalten wieder das Fenster *Optionen*.
   - Bringen Sie das Verzeichnis mit dem Pfeilbutton an die oberste Stelle.
     Dazu markieren Sie zuerst ein beliebiges anderes Verzeichnis, klicken wieder auf *utm-dateiverzeichnis*`\include` und betätigen mehrmals den Pfeilbutton. Ihr Fenster könnte dann z.B. so aussehen.



4. Setzen Sie die Optionen für die UTM-Bibliotheken und die Datenbank-Bibliotheken.

   Dazu wählen Sie in der Box *Verzeichnisse anzeigen für* den Wert *Bibliothekdateien* aus und gehen analog wie bei den Include-Dateien vor:

   a) Verzeichnis mit den UTM-Bibliothekdateien angeben. Diese Option müssen Sie immer setzen:
      – Doppelklicken Sie auf die leere Eingabezeile
      – Doppelklicken Sie auf „ ... “
      – Wählen Sie in *Verzeichnis wählen* das Verzeichnis *utm-dateiverzeichnis*`\sys` aus
      – Drücken Sie OK
      – Bringen Sie das Verzeichnis an die erste Stelle, so dass Sie z.B. folgendes Fenster vor sich haben:



   b) Verzeichnis mit den Datenbanken-Bibliothekdateien angeben. Dies ist nur dann notwendig, wenn Sie eine Datenbank (z.B. ORACLE oder MS-SQL-Server) anschließen wollen:
      – Doppelklicken Sie auf die leere Eingabezeile
      – Doppelklicken Sie auf „ ... “
      – Wählen Sie in *Verzeichnis wählen* das Verzeichnis mit den Datenbank-Bibliotheken aus.
      – Drücken Sie OK

5. Drücken Sie jetzt im Fenster *Optionen* auf OK.
   Damit sind die Optionen für Include-Dateien und Bibliothekdateien gespeichert.

2.2.2 Projekt erzeugen

Rufen Sie das Developer Studio auf und wählen Sie in der Menüleiste den Punkt *Datei*, klicken Sie auf *Neu* und wählen das Blatt *Projekte* aus. Gehen Sie wie folgt vor:

1. Markieren Sie *Win32 Console Application*
2. Schreiben Sie in das Feld *Projektname* den Namen Ihres Projekts; der Name ist frei wählbar.
3. Tragen Sie bei *Pfad* das Verzeichnis ein, in dem Ihr Projekt mitsamt allen Dateien abgelegt werden soll. Wenn Sie hier ein Verzeichnis eintragen, das noch nicht existiert, dann wird es neu angelegt.
4. Klicken Sie OK. Damit wird das Projekt erzeugt, es bleibt dabei geöffnet.

Projekt später öffnen

Wenn Sie das Projekt später öffnen möchten, dann haben Sie zwei Möglichkeiten:

– Über das Developer Studio, indem Sie *Datei - Arbeitsbereich öffnen* klicken und - falls noch nötig - mit Hilfe von *Suchen in* in das Anwendungsverzeichnis navigieren. Dort klicken Sie auf *utmprojekt*`.dsw` (*utmprojekt* = Name Ihres Projekts).

– Über den Explorer, indem Sie auf die Datei *utmprojekt*`.dsw` doppelklicken.

2.2.3 Quellprogramme schreiben

Wenn Sie neue Quellprogramme in C oder C++ erstellen möchten, dann verwenden Sie am besten den syntaxsensitiven Editor des C++ Developer Studios. Sie können Ihre Programme natürlich auch mit einem beliebigem ASCII-Editor erstellen oder vorhandene Programme damit modifizieren.

Wenn Sie Programme mit Visual C++ erstellen, gehen Sie wie folgt vor:

1. Öffnen Sie Ihr Projekt, indem Sie z.B. im Explorer auf die Datei *utmprojekt*`.dsw` doppelklicken. Damit wird das C++ Developer Studio gestartet.
2. Wählen Sie in der Menüleiste den Punkt *Datei*, klicken Sie auf *Neu* und wählen das Blatt *Dateien* aus.
3. Markieren Sie *Textdatei* und tragen Sie bei *Dateiname* den Namen Ihres Quellprogramms ein.
4. Klicken Sie OK. Damit wird diese Datei angelegt und automatisch in das Projekt aufgenommen. Es wird das Editier-Fenster geöffnet.
5. Erstellen Sie im Editier-Fenster den Quellcode. Sie können mit der Maus auch Teile aus anderen Quellcode-Dateien in das Fenster kopieren, indem Sie den Text mit der Maus markieren, die Tasten STRG+C drücken und den Text mit STRG+V in das Fenster einfügen.
6. Schließen Sie die Datei mit *Datei - Schließen* und speichern Sie dabei die Änderungen.

2.2.4 Anwendung übersetzen und binden

Voraussetzungen

Bevor Sie die Anwendung binden können, müssen Sie alle benötigten Quellprogramme und Objektdateien in Ihr Projekt aufnehmen und die Bindeoptionen einstellen, siehe unten. Zu den benötigten Quellprogrammen gehört auch die ROOT-Tabellen-Source, die Sie auf jeden Fall vor dem Binden mit KDCDEF erzeugen müssen. Es wird empfohlen, alle Quellprogramme einschließlich der ROOT-Tabellen-Source im Projektverzeichnis abzulegen.

Quellprogramme und Objektdateien in das Projekt aufnehmen

1. Öffnen Sie Ihr Projekt.
2. Fügen Sie die Quellprogramme wie folgt in Ihr Projekt ein:
   a) Klicken Sie in der Menüleiste auf *Projekt*, markieren *Dem Projekt hinzufügen* und klicken auf *Dateien...* Damit wird das Fenster *Dateien in Projekt einfügen* geöffnet.
   b) Wählen Sie bei *Dateityp* den Punkt *C++-Dateien (.c;.cpp;.cxx;.tli;.h;.tlh;.rc).* Navigieren Sie - falls noch nötig - mit *Suchen in* in das Projektverzeichnis und markieren dort Folgendes:
      – Die mit KDCDEF erzeugte ROOT-Tabellen-Source *rootname*.C
      – Alle selbsterstellten Quellprogramme, die mit eingebunden werden sollen. Quellprogramme, die Sie innerhalb Ihres Projekts mit dem C++-Developer Studio erstellt haben, sind automatisch im Projekt enthalten und brauchen nicht mehr markiert zu werden.
   c) Klicken Sie OK. Damit haben Sie die Quellprogramme in das Projekt aufgenommen.

   Sie können bei umfangreichen Projekten auch einzelne Quellprogramme schon vorher übersetzen (*Erstellen - Komplieren von ...*) und dann stattdessen die Objektdateien hinzufügen wie unten beschrieben.
3. Fügen Sie auf analoge Weise die UTM-Objektdateien zu Ihrem Projekt hinzu:
   a) Klicken Sie in der Menüleiste auf *Projekt*, markieren *Dem Projekt hinzufügen* und klicken auf *Dateien...* Damit wird das Fenster *Dateien in Projekt einfügen* geöffnet.

b) UTM-Objektdateien hinzufügen:
Wählen Sie bei *Dateityp* den Punkt *Objektdateien (.obj)*.
Navigieren Sie bei *Suchen in* auf das Verzeichnis *utm-dateiverzeichnis*`\sys` und markieren folgende Dateien:

- `kcsmsgs.obj` mit den Standard-Meldungsmodulen
- Entweder `mainutm.obj`, wenn keine C++-Programme enthalten sind
- oder `mainutmCC.obj`, wenn C++-Programme enthalten sind.
  `mainutm.obj` bzw. `mainutmCC.obj` enthalten die Main-Funktion der Anwendung.

c) Klicken Sie jetzt auf OK. Damit haben Sie die UTM-Objektdateien eingefügt.

4. Weitere Objektdateien hinzufügen:

   Falls Sie schon Programme übersetzt haben, dann wiederholen Sie die Schritte a) bis c) aus 3. für die zugehörigen Objektdateien.

Dateien anzeigen

Sie können sich jederzeit alle in Ihrem Projekt enthaltenen Dateien anzeigen lassen, indem Sie am unteren Rand des Arbeitsbereichs das Blatt *FileView* anklicken und auf .... *Dateien* doppelklicken. In diesem Fenster können Sie z.B. auch Quellprogramme per Doppelklick öffnen und anschließend editieren.

Binderoptionen einstellen

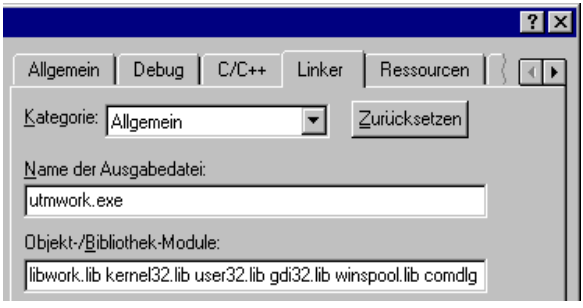
Vor jedem Binden einer Anwendung müssen Sie die Binderoptionen einstellen.

Dazu klicken Sie in der Menüleiste *Projekt - Einstellungen* an, wählen im Feld *Einstellungen für* den Punkt *Alle Konfigurationen* und klicken auf das Blatt *Linker*. In diesem Blatt müssen Sie nacheinander die Optionen für die Kategorien *Allgemein* und *Debug* einstellen. Dabei gehen Sie wie folgt vor:

1. Wählen Sie in der List Box *Kategorie* das Thema *Allgemein* aus und versorgen das Fenster wie folgt:

   a) Tragen Sie bei *Name der Ausgabedatei* den Namen `utmwork.exe` ein.
   Der Name muss immer so lauten, da der Mainprozess von *open*UTM den Workprozesse unter diesem Namen sucht.

   b) Fügen Sie bei *Objekt-/Bibliothek-Module* die UTM-Bibliothek `libwork.lib` **vor** alle anderen Module ein. Diese Bibliothek muss immer angegeben werden, siehe Bildausschnitt:



Falls Sie die XATMI-Schnittstelle oder Datenbanken verwenden, müssen Sie jeweils noch folgende Bibliotheken einfügen:

– Für XATMI die Bibliothek `libcmt.lib`.

– Die Datenbank-Bibliothek(en). Welche Bibliothek(en) angegeben werden müssen, entnehmen Sie bitte der Dokumentation für die jeweilige Datenbank.

c) Aktivieren Sie die Checkbox *Mapfile generieren*.

2. Wählen Sie in der List Box *Kategorie* das Thema *Debug* aus und wählen Sie im Feld *Einstellungen für* den Punkt *Win32 Debug*, da diese Einstellung nur für die Debug-Version sinnvoll ist.

   Aktivieren Sie die Check Box *Debug-Info* und den Radio Button *Beide Formate*.

3. Klicken Sie jetzt OK. Damit sind Ihre gewählten Bindeoptionen gültig.

Hinweis zur exit()-Funktion

Falls Sie in einem Start-Exit die `exit()`-Funktion aufrufen, müssen Sie zusätzlich in der List Box *Kategorie* das Thema *Anpassen* auswählen und die Checkbox *Dateiausgabe erzwingen* aktivieren. Beachten Sie bitte, dass die `exit()`-Funktion nur beim Start-Exit erlaubt ist, in allen anderen Teilprogrammen ist sie verboten.

Anwendungsprogramm binden

Öffnen Sie Ihr Projekt, wählen Sie in der Menüleiste den Punkt *Erstellen* und stellen den Cursor auf den Punkt *utmwork.exe erstellen*, siehe Bildausschnitt:

Sobald Sie anklicken, werden Ihre Programme übersetzt und Ihre Anwendung gebunden.

Meldungen beim Binden

Beim Binden erhalten Sie die Meldungen `LNK4075` und `LNK4056`. Diese Meldungen können Sie ignorieren; die Meldung `LNK4056` entsteht deshalb, weil die Anwendungsprogramme source-kompatibel zu UNIX sind.

Wenn Sie die `exit()`-Funktion in einem Start-Exit verwenden, dann erhalten Sie noch die Meldung `LNK4006`. Ursache dafür ist, dass die `exit()`-Funktion durch die UTM-Bibliothek `libwork.lib` und nicht wie üblich durch eine Windows-Bibliothek abgehandelt wird. Dies ist einer der Gründe, warum `libwork.lib` immer an erster Stelle (s.o.) stehen muss.

Ergebnis des Binderlaufs

Am Ende des Binderlaufs wird das Anwendungsprogramm `utmwork.exe` in Ihrem Projektverzeichnis abgelegt. Da der Mainprozess von *open*UTM das Anwendungsprogramm immer unter diesem Namen sucht, dürfen Sie `utmwork.exe` nicht umbenennen.

2.2.5 Anwendungsprogramme als DLLs erstellen

Wenn Sie ein Anwendungsprogramm als DLL erzeugen möchten, dann gibt es gegenüber dem statischen Binden folgende Abweichungen:

1. Erstellen Sie das Projekt als *Win32 Dynamic-Link Library*, siehe Abschnitt „Projekt erzeugen“ auf Seite 24.
2. Ergänzen Sie alle Teilprogramme um die Anweisung `void declspec( dllexport )`, siehe Abschnitt „Anwendung übersetzen und binden“ auf Seite 26. Dies kann z.B. so aussehen:

   ```
   void declspec( dllexport ) func (struct kc_ca *kb, struct work *spab)
   ```

Alle anderen Schritte sind analog zum statischen Binden.

2.2.6 COBOL-Anwendungsprogramme in Windows

Wenn Sie Anwendungsprogramme in COBOL erstellen möchten, müssen Sie den Compiler NetExpress von Microfocus verwenden.

Anwendungsprogramm übersetzen

Zum Übersetzen sind folgende Schritte notwendig:

1. Setzen Sie die Umgebungsvariable COBCPY auf das Dateiverzeichnis `UTMPATH\copy-cobol85`. Unter diesem Verzeichnis sind die COBOL-Copies abgelegt.

   Falls Sie Clients auf Basis von UPIC-L erstellen, müssen Sie die Dateiverzeichnisse `UTMPATH\upicl\copy-cobol85` bzw. `UTMPATH\xatmi\copy-cobol85` ebenfalls eintragen.
2. Öffnen Sie über *Start - Programme - Eingabeaufforderung* ein DOS-Fenster und geben Sie das Kommando `cobol` ein. Danach geben Sie die Source-Dateien interaktiv an.

   Sie können aber auch das Makefile `workcob.mak` nach eigenen Bedürfnissen anpassen.

   Wenn das Programm mit dem Animator von NetExpress getestet werden soll, ist der Schalter `/ANIM` ist nötig.

Anwendungsprogramm binden

Das Binden erfolgt in zwei Schritten:

1. Öffnen Sie ein DOS-Fenster öffnen und geben Sie das NetExpress-Kommando `cblnames` ein. Dabei geben Sie alle COBOL-Objekte und sonstigen Objekte einzeln an, z.B. `%UTMPATH%\sys\MAINUTMCOB.OBJ` und `root.obj`.
2. Binden Sie das Programm `utmwork.exe` mit Hilfe des Microsoft Binders LINK. Dabei müssen Sie folgende Objekte einbinden:
   - `@cbllds.lnk` (Output von `cblnames`)
   - weitere Anwendugsprogramm-Bibliotheken (falls vorhanden)
   - `%UTMPATH%\sys\libwork.lib` (Import-Bibiothek von UTM)
   - `cblrtss.lib` (Cobol-Laufzeitsystem)
   - C-Laufzeitsystem, z.B. `msvcrt.lib kernel32.lib user32.lib gdi32.lib advapi32.lib`

   Unter Umständen kann es erforderlich sein, die Umgebungsvariable LIB auf die Verzeichnisse mit diesen Bibliotheken zu setzen.

   Die Optionen für beide Schritte übernehmen Sie am einfachsten aus dem QuickStart Kit. Das Makefile für `nmake` wird im Verzeichnis *filebase* unter dem Namen `workcob.mak` abgelegt.

Verwendung von CPIC und XATMI

Wenn COBOL-Programme die CPIC oder XATMI nutzen, müssen sie wie folgt angepasst werden:

1. Es muss ein `SPECIAL-NAMES` Paragraph vor der `DATA DIVISION` eingefügt werden, der die Aufruf-Konvention WINAPI definiert:

   ```
   SPECIAL-NAMES.
   CALL-CONVENTION 74 IS WINAPI.
   ```

2. Jeder Aufruf der CPIC- oder XATMI-Schnittstelle muss mit dieser Konvention erfolgen, z.B. `CALL WINAPI "CMACCP" USING CONVERSATION-ID CM-RETCODE.`
3. Wenn die CPIC benutzt wird, dann müssen die CPIC-Schlüsselwörter `TIMEOUT` und `TIME-OUT` in der Quelldatei CMCOBOL umbenannt werden, da diese Wörter vom Compiler NetExpress reserviert sind. Dazu geben Sie z.B. folgende Anweisung ein:

   ```
   COPY CMCOBOL REPLACING TIME-OUT BY CPIC-TIMEOUT.
   ```

Diese Anmerkungen gelten sowohl für die Verwendung von *open*UTM-Server als auch bei der Verwendung von *open*UTM-Client auf Basis von UPIC (lokale und remote Variante).

Alle Programme des *open*UTM Quick Start Kit sind bereits entsprechend angepasst.

2.2.7 Anwendung als Dienst installieren

Sie können eine vorhandene UTM-Anwendung als Windows-Dienst (Service) einrichten, so dass die Anwendung beim Hochfahren des Rechners automatisch gestartet und beim Herunterfahren automatisch beendet wird. Sie installieren und Deinstallieren einen Dienst, indem Sie das Programm `utmmains` aufrufen. `utmmains` ist Bestandteil von *open*UTM und befindet sich in *utm-dateiverzeichnis*`\ex`.

Installieren eines UTM-Dienstes

Um eine UTM-Anwendung als Dienst zu installieren, müssen Sie sich unter einer Kennung mit Administrationsberechtigung an Windows anmelden und mit *Start - Programme - Eingabeaufforderung* ein Kommandofenster öffnen. Dort rufen Sie `utmmains` wie folgt auf:

`utmmains` [`-d`] `install` *servicename filebase* [*startparam-file* [*user*]]

Die Parameter müssen durch Leerzeichen getrennt sein und haben folgende Bedeutung:

-d Es werden zusätzlich Diagnoseinformationen auf *stdout* ausgegeben; optional.

install gibt an, dass `utmmains` einen Dienst installiert.

servicename
: Variabler Namensteil des Dienstes; Pflichtparameter.
Der vollständige Name des Dienstes lautet `openUTM` *servicename*. Es wird empfohlen, für *servicename* den Anwendungsnamen aus der KDCDEF-Anweisung MAX APPLINAME zu nehmen.

filebase
: Vollqualifizierter Basisname der UTM-Anwendung; Pflichtparameter.
Der Basisname ist der Name des Verzeichnisses, in dem sich Anwendungsprogramm und KDCFILE befinden.

startparam-file
: Vollqualifizierter Name der Startparameterdatei.

  Standard: Startp.std

user Konto-Name für den Ablauf der Anwendung.
Hier können Sie einen lokalen Benutzer oder einen Domänen-Benutzer angeben. Der Name muss folgende Syntax besitzen:

  Bei einem lokalen Benutzerkonto: *.\LocalUser*
  Bei einem Domänen-Benutzerkonto: *DomainName\DomainUser*
  Standard: Systemkonto

  Wenn Sie hier einen Benutzer angeben, dann müssen Sie das zugehörige Passwort beim Konfigurieren des Dienstes eintragen, siehe unten.

`utmmains` richtet den Dienst sofort ein. Danach müssen Sie den Dienst konfigurieren.

Konfigurieren des UTM-Dienstes

Als Administrator können Sie die Startart und die Eigenschaften des Anmelde-Kontos festlegen. Dazu gehen Sie wie folgt vor:

1. Öffnen Sie das Fenster *Systemsteuerung*, indem Sie *Start - Einstellungen - Systemsteuerung* klicken.
2. Klicken Sie auf *Dienste*. Damit wird ein Fenster geöffnet, das alle auf dem Rechner vorhandenen Dienste auflistet.
3. Markieren Sie den gewünschten Dienst; UTM-Dienste haben immer das Präfix `openUTM`. Die Spalte *Status* zeigt an, ob der Dienst gestartet ist oder nicht.
4. Klicken Sie auf den Button *Startart...*. Damit wird ein weiteres Fenster geöffnet, in dem Sie Folgendes festlegen:
   - Die Startart (Manuell/Automatisch/Deaktiviert)
   - Das Konto, unter dem der Dienst angemeldet wird.
     Wenn Sie beim Installieren des Dienstes schon einen Benutzer angegeben haben (Parameter *user*), wird er hier angezeigt. In diesem Fall **müssen** Sie hier das zugehörige Passwort eingeben, sonst können Sie den Dienst nicht starten.

   i Wenn ein Dienstprogramm wie z.B. *KDCKAA.exe* nicht auf die Ressourcen der UTM-Anwendung (z.B. Shared Memory) zugreifen kann , dann kann das daran liegen, dass es sich um unterschiedliche Konten handelt. In diesem Fall sollten Sie den Dienst so konfigurieren, dass er mit den gleichen Kontodaten angemeldet wird wie der Benutzer, der das Dienstprogramm startet.
5. Klicken Sie OK und verlassen Sie die Systemsteuerung.

Deinstallieren eines UTM-Dienstes

Um einen UTM-Dienst zu deinstallieren, melden Sie sich unter einer Kennung mit Administrationsberechtigung an, öffnen ein Fenster *Eingabeaufforderung* und rufen `utmmains` mit folgenden Parametern auf:

`utmmains [-d] remove` ***servicename***

Bedeutung der Parameter:

-d Es werden zusätzlich Diagnoseinformationen auf *stdout* ausgegeben; optional.

remove
gibt an, dass `utmmains` einen Dienst deinstalliert.

servicename
Variabler Namensteil des UTM-Dienstes.

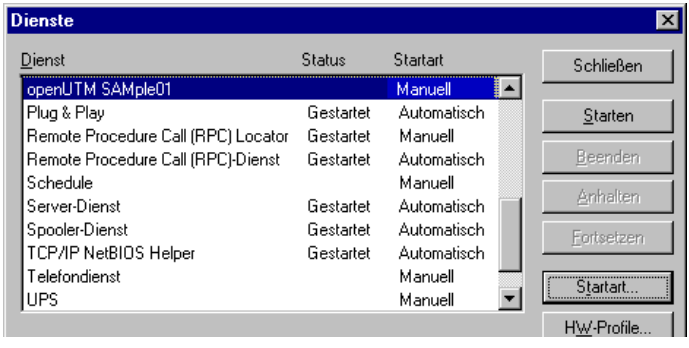
Beispiel

Sie möchten die Anwendung `SAMple01` als Dienst einrichten und auf das Systemkonto anmelden. KDCFILE, Startparameterdatei und Anwendungsprogramm befinden sich im Verzeichnis `C:\utmserv`. Die Startparameterdatei besitzt den Namen `startparameter`.

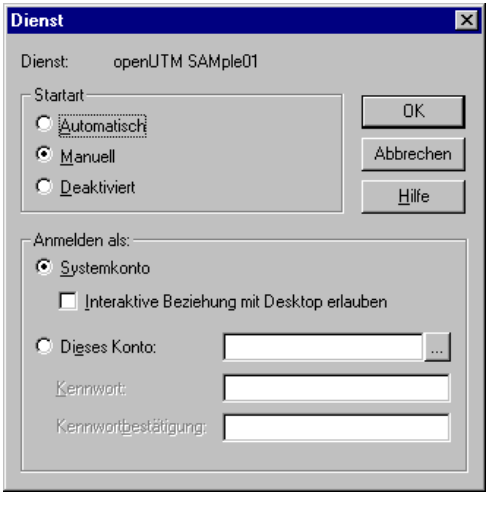
- Um den Dienst zu installieren, rufen Sie `utmmains` wie folgt auf:

  ```
  utmmains install SAMple01 C:\utmserv C:\utmserv\startparameter
  ```

  Wenn Sie jetzt nacheinander *Start - Einstellungen - Systemsteuerung - Dienste* anklicken und nach unten blättern, siehe z.B. folgendes Bild:



  Klicken Sie auf *Startart...* und konfigurieren Sie den Dienst in folgendem Fenster:



  Hier können Sie z.B. bei Startart auf *Automatisch* umschalten oder bei *Anmelden an* an Stelle des Systemkontos bei *Dieses Konto* ein anderes Konto angeben.

- Zum Deinstallieren rufen Sie `utmmains` wie folgt auf:

  ```
  utmmains remove SAMple01
  ```

3 Notwendige Dateien und systemglobale Betriebsmittel

Vor jedem Starten einer UTM-Anwendung müssen Sie dafür sorgen, dass die folgenden für den Betrieb einer UTM-Anwendung notwendigen Dateien vorhanden sind:

- die KDCFILE
- die System-Protokolldatei SYSLOG
- die Benutzer-Protokolldatei(en) USLOG (optional)
- das Dateiverzeichnis DUMP
- alle Programm- und Objektmodul-Bibliotheken, aus denen beim Starten und während des Betriebs der Anwendung Module dynamisch nachgeladen werden sollen.

Diese Dateien bzw. Dateiverzeichnisse müssen im Dateiverzeichnis *filebase* stehen. Den Namen *filebase* legen Sie beim Erzeugen der KDCFILE mit dem Generierungstool KDCDEF fest, siehe *open*UTM-Handbuch „Anwendungen generieren“. *filebase* geben Sie mit der Steueranweisung MAX...,KDCFILE=*filebase* an.

3.1 System-Protokolldatei SYSLOG

*open*UTM protokolliert alle Ereignisse aus dem Lauf der Anwendung in die System-Protokolldatei SYSLOG (**SYS**TEM **LOG**GING), d.h. *open*UTM schreibt alle Meldungen mit Meldungsziel SYSLOG in diese Datei (zu „Meldungsziel“ siehe *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“). Dabei arbeitet *open*UTM mit Wechselpuffern. Dies vermeidet Wartesituationen und verbessert so die Performance, vor allem in Anwendungen mit einem hohen Aufkommen an SYSLOG-Meldungen.

Die System-Protokolldatei SYSLOG können Sie für die laufende Überwachung des Anwendungslaufs oder für spätere Kontrollen nutzen. Insbesondere für die Diagnose liefert die SYSLOG wichtige Information.

Die SYSLOG der Anwendung liegt immer unter dem Dateiverzeichnis *filebase,* wobei *filebase* das Dateiverzeichnis ist, unter dem die Anwendung installiert ist (Basisname der KDCFILE).

*open*UTM bietet zwei Möglichkeiten, eine SYSLOG zu führen:

– als einfache Datei SYSLOG im Basisverzeichnis *filebase*

– als Dateigenerationsverzeichnis (**F**ile **G**eneration **G**roup, kurz FGG) SYSLOG im Basisverzeichnis *filebase*

Gegenüber einer einfachen SYSLOG-Datei hat eine SYSLOG-FGG folgende Vorteile:

- Sie können im laufenden Betrieb der Anwendung auf die jeweils folgende Dateigeneration umschalten (umschaltbare SYSLOG-Datei). Die SYSLOG können Sie z.B. mit dem Administrationskommando KDCSLOG administrieren. Sehen Sie dazu *open*UTM-Handbuch „Anwendungen administrieren“. Beim Umschalten schließt *open*UTM die zuvor beschriebene Dateigeneration.
- Sie können eine automatische Größenüberwachung für die SYSLOG einstellen. D.h. Sie können einen Schwellwert für die Größe der einzelnen Dateigenerationen der SYSLOG-FGG generieren bzw. per Administration festlegen, bei dem *open*UTM automatisch auf die folgende Dateigeneration der FGG umschaltet. Die Größenüberwachung kann während des Anwendungsbetriebs ein- und ausgeschaltet werden.

Meldungen von *open*UTM

*open*UTM gibt die folgenden Meldungen bezüglich der SYSLOG aus:

- Beim Anwendungsstart die Meldung K136:

```
K136 (Erste) SYSLOG-Datei ist &FNAM
```

- Beim Anwendungsende oder nach Umschalten auf eine andere Dateigeneration die Meldung K138:

```
K138 SYSLOG-Datei &FNAM geschlossen
```

3.1.1 SYSLOG als einfache Datei

Sie können die Datei SYSLOG vor dem Start der Anwendung selbst als einfache Datei im Basisverzeichnis *filebase* anlegen. Existiert beim Start der Anwendung unter dem Namen SYSLOG weder eine Datei noch ein Dateigenerationsverzeichnis in *filebase*, dann legt *open*UTM eine einfache Datei mit dem Namen SYSLOG an.

Beim Start der Anwendung öffnet *open*UTM die Datei SYSLOG. Sie bleibt während des gesamten Anwendungslaufs geöffnet. In sie schreibt *open*UTM alle Ereignisse eines Anwendungslaufs.

Bei jedem Folgestart der Anwendung wird der Inhalt der SYSLOG-Datei von *open*UTM überschrieben. Die Protokollinformation aus dem vorherigen Anwendungslauf geht verloren. Nach dem Ende eines Anwendungslaufs sollten Sie deshalb, falls nötig, den Inhalt der SYSLOG-Datei sichern.

ACHTUNG!

Wollen Sie die SYSLOG als einfache Datei führen, dann dürfen Sie die Größenüberwachung für die SYSLOG **nicht** generieren. Wenn Sie in diesem Fall die Größenüberwachung bei der Generierung einschalten mit MAX...,SYSLOG-SIZE=*size* (*size* > 0), dann bricht *open*UTM den Start der Anwendung mit Startfehlercode 58 ab.

3.1.2 SYSLOG als Dateigenerationsverzeichnis

*open*UTM führt die SYSLOG nur als Dateigenerationsverzeichnis, wenn beim Start der Anwendung im Basisverzeichnis *filebase* ein Dateigenerationsverzeichnis mit dem Namen SYSLOG existiert. Dieses Dateigenerationsverzeichnis müssen Sie mit dem Tool KDCSLOG anlegen, siehe Abschnitt „Das Tool KDCSLOG zum Anlegen der SYSLOG-FGG“ auf Seite 38.

Ein Dateigenerationsverzeichnis FGG (**F**ile **G**eneration **G**roup) ist ein Dateiverzeichnis mit Dateien, die über ihren Dateinamen durchnummeriert sind (z.B. 0001, 0002,...). Die Dateien heißen Dateigenerationen der FGG. Die Nummern heißen Dateigenerationsnummern.

Beim Einrichten der FGG mit KDCSLOG wird die Datei INFO erzeugt und in die FGG geschrieben. Beim ersten Start der Anwendung legt *open*UTM die erste Dateigeneration 0001 in der FGG an und öffnet sie als SYSLOG-Datei. Alle Prozesse der Anwendung schreiben die Meldungen mit Ziel SYSLOG zunächst in diese Dateigeneration.

Beim Umschalten legt *open*UTM die Folgegeneration selbst an.

Ist die Dateigeneration mit der Generationsnummer *n* die letzte Datei, in die *open*UTM vor dem Ende eines Anwendungslaufs geschrieben hat, dann erzeugt *open*UTM beim folgenden Anwendungsstart die Dateigeneration (*n*+1) und öffnet sie als SYSLOG-Datei.

In der FGG sind maximal *m* Dateigenerationen enthalten. Die Zahl *m* legen Sie beim Erzeugen der FGG mit dem Tool KDCSLOG fest. Sobald *open*UTM die (*m*+1)-te Dateigeneration anlegt, wird die älteste Dateigeneration gelöscht, d.h. die Dateigeneration mit der niedrigsten Generationsnummer.

3.1.3 Das Tool KDCSLOG zum Anlegen der SYSLOG-FGG

Die SYSLOG-FGG richten Sie mit dem Tool KDCSLOG ein. Sie finden es im Unterverzeichnis `ex` des Installationsverzeichnisses von *open*UTM. Es wird wie folgt gestartet:

In UNIX aus der Shell mit

utm-dateiverzeichnis`/ex/kdcslog` *filebase number* `[K]`

In Windows in einem DOS-Fenster mit

utm-dateiverzeichnis`\ex\kdcslog` *filebase number* `[K]`

Bedeutung der Parameter:

filebase — Name des Dateiverzeichnisses, unter dem die Anwendung installiert ist bzw. installiert werden soll (Basisname der KDCFILE).

number — Maximale Anzahl der Dateigenerationen in der FGG.
In der FGG sind maximal *number* Dateigenerationen enthalten. Sobald *open*UTM die (*number*+1)-te Dateigeneration anlegt, wird die älteste Dateigeneration gelöscht (d.h. die Dateigeneration mit der niedrigsten Generationsnummer).

Minimalwert: 1
Maximalwert: 9999

K — (keep)
Wenn dieser Parameter angegeben wird, werden alle Dateien aufgehoben, auch wenn *number* überschritten wird.

KDCSLOG legt zunächst das Basisverzeichnis *filebase* an, falls es noch nicht existiert. Anschließend wird unter *filebase* die FGG SYSLOG eingerichtet und innerhalb der FGG eine INFO-Datei angelegt. In der Datei INFO sind aktuelle Statusinformationen über Dateigenerationen der Gruppe abgelegt.

ACHTUNG!

Existiert im Verzeichnis *filebase* vor dem Aufruf von KDCSLOG bereits eine SYSLOG FGG, so wird diese FGG gelöscht und eine neue eingerichtet.

Meldungen von KDCSLOG

Das Tool KDCSLOG gibt seine Meldungen nach *stdout* und *stderr* aus. Die Meldungen von KDCSLOG finden Sie im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

Beispiel

FGG SYSLOG erzeugen unter UNIX

*open*UTM ist installiert im Dateiverzeichnis `/opt/lib`, der Basisname der Anwendung ist `/home/userutm/beispiel`. Das Dateigenerationsverzeichnis für die SYSLOG richten Sie wie folgt ein:

```
/opt/lib/utm52a00/ex/kdcslog /home/userutm/beispiel 10
```

KDCSLOG erzeugt die FGG:

```
/home/userutm/beispiel/SYSLOG
```

und die Datei:

```
/home/userutm/beispiel/SYSLOG/INFO
```

FGG SYSLOG erzeugen unter Windows

*open*UTM ist installiert im Dateiverzeichnis `C:\openUTM-Server`, der Basisname der Anwendung ist `C:\utmsample`. Das Dateigenerationsverzeichnis für die SYSLOG richten Sie wie folgt ein:

```
C:\openUTM-Server\ex\kdcslog C:\utmsample 10
```

KDCSLOG erzeugt die FGG:

```
C:\utmsample\SYSLOG
```

und die Datei:

```
C:\utmsample\SYSLOG\INFO
```

Anmerkungen zu den Beispielen

Die UTM-Anwendung schreibt immer in die Datei mit der aktuell höchsten Generationsnummer. Wird die SYSLOG auf die nächste Dateigeneration umgeschaltet, dann legt *open*UTM diese Dateigeneration an. Es kann höchstens soviel nummerierte Protokolldateien geben, wie für den Parameter *number* angegeben wurde, d.h. maximal 10 Dateigenerationen. Ist die Anzahl erreicht und wird umgeschaltet, so wird die Datei mit der niedrigsten Nummer gelöscht, d.h wenn *open*UTM beim Umschalten die Dateigeneration 0011 anlegt, wird automatisch die Generation 0001 gelöscht usw.

ACHTUNG!

Achten Sie darauf, dass noch nicht ausgewertete Dateien nicht überschrieben bzw. gelöscht werden.

3.1.3.1 Automatische Größenüberwachung

Die automatische Größenüberwachung können Sie nur für FGGs verwenden. Wenn Sie die SYSLOG-Datei als einfache Datei anlegen und die automatische Größenüberwachung generieren, dann bricht *open*UTM den Start der Anwendung mit Startfehlercode 58 ab.

Sie können die automatische Größenüberwachung auf zwei Arten einstellen:

- bei der Generierung mit der KDCDEF-Anweisung MAX ...,SYSLOG-SIZE=*size*
- im laufenden Betrieb der Anwendung mit dem Administrationskommando KDCSLOG [SWITCH,]SIZE=*size* oder an der Programmschnittstelle zur Administration mit dem Operationscode KC_SYSLOG und Subopcode KC_CHANGE_SIZE (siehe *open*UTM-Handbuch „Anwendungen administrieren“)

In beiden Fällen müssen Sie für *size* einen Wert > 0 angeben.

Bei eingeschalteter Größenüberwachung überprüft *open*UTM vor jeder Meldungsausgabe in die SYSLOG-Datei, ob durch die Meldungsausgabe die vereinbarte Maximalgröße der Dateigeneration (*size* ∗ Größe einer UTM-Seite) überschritten würde. Ist dies der Fall, dann wird vor der Meldungsausgabe versucht, auf die nächste Dateigeneration zu schalten. Bei Erfolg gibt *open*UTM die Meldung K137 aus. Die Meldung wird in die neue Dateigeneration geschrieben.

Tritt bei dem Versuch umzuschalten ein Fehler auf, dann arbeitet *open*UTM mit der alten Dateigeneration weiter, in die vor dem Umschaltversuch protokolliert wurde. *open*UTM schreibt die Meldung K139 auf *stdout* und auf die Konsole des Administrators. Außerdem wird wie bei allen DMS-Fehlern zusätzlich die Meldung K043 ausgegeben. Sie enthält einen DMS-Fehlercode, dem Sie den Grund für den Umschaltfehler entnehmen können.

Damit *open*UTM nicht bei jeder folgenden Meldung mit dem Ziel SYSLOG erneut erfolglos versucht, auf die nächste Dateigeneration umzuschalten, wird die automatische Größenüberwachung nach einem solchen Umschaltfehler suspendiert.

Nachdem der Administrator den Grund für den Umschaltfehler gefunden und beseitigt hat, kann er die automatische Größenüberwachung wieder aktivieren, z.B. mit dem Kommando KDCSLOG SWITCH. Mit dem KDCSLOG SWITCH zwingt er *open*UTM, einen erneuten Umschaltversuch zu starten. Verläuft dieser Versuch ohne Fehler, so wird eine vorher suspendierte Größenüberwachung automatisch wieder aktiviert.

Nach dem erfolgreichen Umschalten werden keine Meldungen mehr in die alte Dateigeneration geschrieben. Wird eine Dateigeneration geschlossen, so gibt *open*UTM die Meldung K138 aus.

3.1.4 Schutz vor zu großer SYSLOG-Datei

Sie können die Belegung von Speicherplatz durch die SYSLOG kontrollieren, wenn Sie die SYSLOG als FGG führen, für die FGG maximal *n* Dateigenerationen zulassen (Parameter *number* des Tools KDCSLOG) und für die SYSLOG die automatische Größenüberwachung einschalten. Sehen Sie dazu Abschnitt „Automatische Größenüberwachung“ auf Seite 40.

Die Dateigenerationen der SYSLOG werden zyklisch überschrieben, so dass die FGG maximal *n* Dateigenerationen enthält. Jede Generation hat durch die Größenüberwachung eine maximale Größe von *size* UTM- Seiten.

Damit ist der maximale Speicherverbrauch der SYSLOG-FGG:

n ∗ *size* ∗ (Größe einer UTM-Seite)

3.1.5 Verhalten bei Schreibfehlern

Tritt bei dem Versuch, eine Meldung in die SYSLOG zu schreiben, ein Fehler auf, dann gibt *open*UTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen.

Das weitere Vorgehen von *open*UTM ist abhängig davon, ob die SYSLOG als einfache Datei oder als FGG geführt wird.

- Die SYSLOG wird als einfache Datei geführt

  Nach Ausgabe der Meldung K043 wird die Anwendung mit Grund SLOG09 abgebrochen.

- Die SYSLOG wird als FGG geführt

  *open*UTM versucht beim Auftreten eines Fehlers auf die nächste Dateigeneration zu schalten. *open*UTM schaltet auch um, wenn die Größenüberwachung ausgeschaltet bzw. nicht generiert ist. *open*UTM schaltet nicht um, wenn die Größenüberwachung auf Grundrund eines vorangegangenen Umschaltfehlers suspendiert ist.

  Schlägt der Umschaltversuch fehl, dann wird die Anwendung mit Grund SLOG09 abgebrochen.

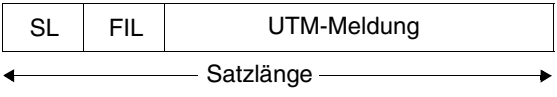
  Kann *open*UTM auf die nächste Dateigeneration umschalten, dann versucht *open*UTM erneut die Meldung in die SYSLOG zu schreiben. Tritt dabei ein Fehler auf, wird die Anwendung mit SLOG09 abgebrochen. Tritt kein Fehler auf, läuft die Anwendung weiter, *open*UTM protokolliert in die neue SYSLOG-Dateigeneration.

3.1.6 SYSLOG-Datei auswerten

*open*UTM erstellt die SYSLOG-Datei in einem Format, das nicht direkt auswertbar ist. Um sie lesbar ausdrucken oder mit einem Programm bearbeiten zu können, muss man sie aufbereiten. Dazu gibt es die zwei Aufbereitungstools: KDCCSYSL und KDCPSYSL.

3.1.6.1 Das Tool KDCCSYSL - SYSLOG-Datei aufbereiten

Das Aufbereitungstool KDCCSYSL wandelt eine SYSLOG-Datei in eine andere Datei um, die mit einem benutzereigenen Auswerteprogramm verarbeitet werden kann. Jede Meldung dieser neuen Datei besteht aus einem so genannten Satz variabler Länge, d.h. eine Meldung ist in folgendem Format abgelegt:



SL 2 byte; Inhalt: Satzlänge binär (Länge der Meldung + 4)

FIL 2 byte; Inhalt: Leerzeichen

Der Aufbau der einzelnen UTM-Meldungen entspricht den Strukturen im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

Aufruf von KDCCSYSL

UNIX:

utm-dateiverzeichnis`/ex/kdccsysl` *file_in file_out*

Windows in einem DOS-Fenster:

utm-dateiverzeichnis`\ex\kdccsysl` *file_in file_out*

Bedeutung der Parameter:

file_in ist hierbei die SYSLOG-Datei

file_out ist die von KDCCSYSL aufbereitete SYSLOG-Datei

Die Meldungen von KDCCSYSL werden nach *stderr* ausgegeben.

3.1.6.2 Das Tool KDCPSYSL - Meldungstexte erzeugen

KDCPSYSL liest aus der mit dem Aufbereitungstool KDCCSYSL erzeugten Datei und erzeugt eine weitere Datei. Diese enthält zu jeder Meldung die Meldungsnummer und den Meldungstext mit den aktuellen Textparametern, Datum und Uhrzeit sowie zusätzliche aktuelle Parameter, die nicht im Meldungstext enthalten sind. Einige Inserts werden abdruckbar ausgegeben.

Die mit KDCPSYSL erzeugte Datei ist eine normale Textdatei, die pro Meldung eine oder mehrere Zeilen enthält. Diese Datei, die z.B. zur schnellen Übersicht dienen kann, können Sie mit einem Editor ansehen und evtl. weiter verarbeiten oder ausdrucken. Die Sätze der Datei haben eine Länge von bis zu 136 Zeichen (Spalten). Dies ist beim Ausdrucken zu berücksichtigen.

Aufruf von KDCPSYSL

UNIX:

utm-dateiverzeichnis`/ex/kdcpsysl`

Windows aus einem DOS-Fenster:

utm-dateiverzeichnis`\ex\kdcpsysl`

KDCPSYSL arbeitet mit den festen Dateinamen `slogin` für die mit KDCCSYSL erzeugte Eingabedatei und `slogout` für die aufbereitete Ausgabedatei. Gegebenenfalls ist daher vor dem Aufruf von KDCPSYSL die Datei *file_out* vom KDCCSYSL-Aufruf in `slogin` umzubenennen.

KDCPSYSL bereitet die Daten in `slogin` mit Hilfe von NLS in der durch die Umgebungsvariable LANG eingestellten Sprache auf, sofern unter
utm-dateiverzeichnis`/nls/msg/`*eingestellte-Sprache* (UNIX) bzw.
utm-dateiverzeichnis`\nls\msg\`*eingestellte-Sprache* (Windows) ein Meldungskatalog `utmsys.cat` existiert. Andernfalls wird der eingebundene Meldungskatalog verwendet.

3.1.6.3 Meldungen von KDCCSYSL und KDCPSYSL

Die Aufbereitungstools KDCCSYSL und KDCPSYSL schreiben ihre Meldungen nach *stderr*. KDCCSYSL und KDCPSYSL bereiten ihre Meldungen mit Hilfe von NLS in der durch die Umgebungsvariable LANG eingestellten Sprache auf, sofern unter
utm-dateiverzeichnis`/nls/msg/`*eingestellte-Sprache* (UNIX) bzw.
utm-dateiverzeichnis`\nls\msg\`*eingestellte-Sprache* (Windows) ein NLS-Meldungskatalog `utmsys.cat` existiert. Andernfalls wird der eingebundene Meldungskatalog verwendet.

Die Meldungen finden Sie im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

3.2 Benutzer-Protokolldatei

In der Benutzer-Protokolldatei stehen die Sätze, die das Anwendungsprogramm mit LPUT-Aufrufen erzeugt hat. Die Benutzer-Protokolldateien einer Anwendung sind in einem Dateigenerationsverzeichnis FGG (**F**ile **G**eneration **G**roup) organisiert, d.h. einer Gruppe von Dateien, die über ihren Dateinamen durchnummeriert sind. Die Benutzer-Protokoll-Dateien stehen im Dateiverzeichnis USLA unter dem Basisverzeichnis *filebase*. Wenn Benutzer-Protokoll-Dateien benötigt werden (LPUT-Aufrufe), dann müssen diese vor dem Anwendungsstart mit dem Tool KDCUSLOG eingerichtet werden.

Aufruf von KDCUSLOG

UNIX:

utm-dateiverzeichnis`/ex/kdcuslog` *filebase number* `[ S | D ]`

Windows aus der DOS-Shell:

utm-dateiverzeichnis`\ex\kdcuslog` *filebase number* `[ S | D ]`

Bedeutung der Parameter:

filebase	Name des Verzeichnisses, unter dem die Anwendung installiert ist oder installiert werden soll; Basisname der KDCFILE.
number	Zahl der Dateien pro Dateigenerationsverzeichnis; maximal 9999.
S	einfache Dateiführung; Standardeinstellung.
D	doppelte Dateiführung; es wird zusätzlich das Dateiverzeichnis USLB im Basisverzeichnis *filebase* erzeugt.

KDCUSLOG legt zunächst das Dateiverzeichnis *filebase* an, wenn es noch nicht existiert. Anschließend werden das Dateiverzeichnis USLA und bei doppelter Dateiführung zusätzlich USLB in *filebase* eingerichtet. Innerhalb des Dateiverzeichnisses USLA bzw. USLB wird eine INFO-Datei angelegt, in der aktuelle Statusinformationen über Dateien der FGG hinterlegt werden.

Das Dateiverzeichnis USLA enthält folgende Dateien:

/INFO	Verwaltungsdatei
0001	erste Datei der Dateigeneration (Generationsnummer 0001)

ACHTUNG!

Existiert vor dem Aufrufen der Prozedur bereits das Dateigenerationsverzeichnis, so wird die alte Gruppe gelöscht und eine neue eingerichtet.

Beispiel

UNIX

*open*UTM ist installiert im Dateiverzeichnis `/opt/lib`, der Basisname der Anwendung ist `/home/userutm/beispiel`. Die Benutzer-Protokolldatei soll doppelt geführt werden.

Das Dateigenerationsverzeichnis für die Benutzer-Protokolldatei richten Sie wie folgt ein:

```
/opt/lib/utm52a00/ex/kdcuslog /home/userutm/beispiel 2 D
```

KDCUSLOG erzeugt dann die Dateien:

```
/home/userutm/beispiel/USLA/INFO
/home/userutm/beispiel/USLA/0001

/home/userutm/beispiel/USLB/INFO
/home/userutm/beispiel/USLB/0001
```

Windows

*open*UTM ist installiert im Dateiverzeichnis `C:\openUTM-Server`, der Basisname der Anwendung ist `C:\utmsample`. Die Benutzer-Protokolldatei soll doppelt geführt werden.

Das Dateigenerationsverzeichnis für die Benutzer-Protokolldatei richten Sie wie folgt ein:

```
C:\openUTM-Server\ex\kdcuslog C:\utmsample 2 D
```

KDCUSLOG erzeugt dann die Dateien:

```
C:\utmsample\USLA\INFO
C:\utmsample\USLA\0001

C:\utmsample\USLB\INFO
C:\utmsample\USLB\0001
```

Anmerkungen zum Beispiel

Die UTM-Anwendung schreibt immer in die Datei mit der aktuell höchsten Nummer. Bei jedem KDCLOG-Kommando durch den Administrator wird auf die folgende Dateigeneration weitergeschaltet. Es kann höchstens soviel nummerierte Benutzer-Protokolldateien geben, wie unter dem Parameter *number* (im Beispiel 2) beim Aufruf von KDCUSLOG angegeben wurden. Ist die Anzahl erreicht und wird mit KDCLOG weitergeschaltet, so wird die Datei mit der niedrigsten Nummer gelöscht.

Achten Sie darauf, dass noch nicht ausgewertete Dateien nicht überschrieben werden.

Die Benutzer-Protokoll-Sätze schreibt *open*UTM nicht sofort in die Protokolldatei, sondern speichert sie erst im Pagepool der KDCFILE. Sind im Pagepool so viele UTM-Seiten belegt, wie in MAX...,LPUTBUF=*number* generiert, kopiert *open*UTM die Sätze in die Benutzer-Protokolldatei. Das Kopieren erfolgt asynchron zu laufenden Transaktionen. Beim normalen Beenden der Anwendung kopiert *open*UTM die Sätze ebenfalls in die Benutzer-Protokolldatei.

Die Anzahl der bei LPUTBUF=*number* angegebenen UTM-Seiten ist bei der Generierung der Größe des Pagepools mit MAX...,PGPOOL=*number* zu berücksichtigen.

Mit MAX...,LPUTLTH=*length* beeinflussen Sie die Blocklänge der Benutzer-Protokolldatei. Sie wird von *open*UTM berechnet und kann größer sein als die Standardblockung 2KB.

*open*UTM kann LPUT-Sätze nur in die Benutzer-Protokolldatei kopieren, wenn diese eingerichtet ist und *open*UTM darauf zugreifen kann.

Bitte beachten Sie, dass die Benutzer-Protokolldatei nach einem KDCDEF-Lauf oder nach einem KDCUPD-Lauf von Anfang an überschrieben wird; ansonsten wird sie fortgeschrieben. Deshalb sollten Sie die Protokoll-Sätze vor einem KDCDEF- bzw. KDCUPD-Lauf auswerten.

Meldungen von KDCUSLOG

KDCUSLOG gibt seine Meldungen nach *stdout* und *stderr* aus. Die Meldungen von KDCUSLOG finden Sie im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

3.2.1 Verhalten bei Schreibfehlern

Tritt beim Schreiben von LPUT-Sätzen in die Benutzer-Protokolldatei ein DMS-Fehler (**D**ata **M**anagement **S**ystem) auf, so gibt *open*UTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen. Gleichzeitig wird jeder weitere LPUT-Aufruf im Teilprogramm mit dem KDCS-Returncode 40Z (interner Returncode K903) abgewiesen.

Der Administrator der Anwendung kann dann die Benutzer-Protokolldatei bzw. ihre Generationen korrigieren, restaurieren oder neu einrichten.

Damit *open*UTM wieder LPUT-Sätze in die Benutzer-Protokolldatei schreibt, muss der Administrator der Anwendung nach der Fehlerbehebung das Administrationskommando KDCLOG oder einen KDCADMI-Aufruf mit Opcode KC_USLOG absetzen (siehe *open*UTM-Handbuch „Anwendungen administrieren“).

Die im Pagepool der KDCFILE gesicherten LPUT-Sätze werden jetzt in die Protokolldatei(en) geschrieben, die Dateigenerationsnummer wird erhöht.
Die Sperre für die LPUT-Aufrufe in den Teilprogrammen wird aufgehoben.

3.3 Dateiverzeichnis DUMP

In dem Dateiverzeichnis DUMP werden folgende Dateien abgelegt:

– Dump-Dateien, die evtl. während des Anwendungslaufs erzeugt wurden

– Temporäre Dateien, die für die Dump-Erstellung notwendig sind

– Bei Bedarf auch core-Dateien für die Diagnose

Dieses Dateiverzeichnis sollten Sie daher stets vor dem Start einrichten, damit diese Dateien erzeugt werden können. Das Dateiverzeichnis DUMP müssen Sie im Basisverzeichnis *filebase* anlegen.

3.4 Systemglobale Betriebsmittel einer Anwendung

In diesem Abschnitt werden die systemglobalen Betriebsmittel aufgelistet, die eine *open*UTM-Produktivanwendung benötigt. Sie erfahren, wie Sie die Größe des Shared-Memory-Bereichs für die Inter-Prozess-Kommunikation (IPC) anpassen können, um die Performance Ihrer Anwendung bei der Kommunikation mit Netz-Partnern zu verbessern.

3.4.1 Von einer UTM-Anwendung benötigte Betriebsmittel

Eine *open*UTM-Produktivanwendung benötigt die im Folgenden aufgeführten systemglobalen Betriebsmittel.

Shared Memory-Bereiche

Eine UTM-Anwendung benötigt drei Shared Memory-Bereiche für die Konfigurations- und anwendungsglobalen Verwaltungsdaten (KAA), den Cache und die UTM-interne Prozesskommunikation (siehe auch Abschnitt „Performance-Verbesserung: Größe der Datenbereiche im IPC-Shared-Memory anpassen“ auf Seite 50).

Semaphore

Eine UTM-Anwendung benötigt Semaphore für die Realisierung von Wartesituationen („Warteschlangen, Message Queues“). Semaphore sind Variablen zur Steuerung und Synchronisation von Workprozessen.

In *open*UTM sind die Semaphore in so genannten Semaphor-Arrays organisiert, wobei jedes Semaphor-Array genau 20 Semaphor-Einträge enthält. Eine UTM-Anwendung benötigt mindestens ein Semaphor-Array. Die maximale Anzahl der Semaphore im System ist beschränkt.

In der *open*UTM-Umgebung werden die Semaphor-Einträge folgendermaßen belegt:

- acht Einträge für IPC-Shared-Memory
- ein Eintrag für KAA-Shared-Memory
- ein Eintrag für CACHE-Shared-Memory
- ein Eintrag für CACHE-Zugriffssperre
- ein Eintrag für PCMM-Zugriffssperre
- ein Eintrag für „UTM-Workbörse“
- je ein Eintrag für jeden Workprozess als Tasklockbörse
- je ein Eintrag für jeden angeschlossenen externen Prozess (*utmtimer*, *utmnet*, *utmnets*, *utmdtp*, *utmnetm*, *utmnetc*, *utmprint* und lokales UPIC-Client-Programm) zur Kommunikation zwischen *open*UTM und externem Prozess.

  Wieviele externe Prozesse gestartet werden, ist abhängig von der Art der Netz-Anbindung einer UTM-Anwendung:

  Anbindung über CMX/PCMX

  – Die Anbindung ans Netz erfolgt multi-threaded (Standard):
    (MAX ...,NET-ACCESS=MULTI-THREADED)

    Für jede Listener-ID, die mit KDCDEF generiert wird (BCAMAPPL- oder ACCESS-POINT-Anweisung), wird je ein Netzprozess (*utmnet*) gestartet.

  – Die Anbindung ans Netz erfolgt single-threaded (UNIX):
    (MAX ...,NET-ACCESS=SINGLE-THREADED)

    In diesem Fall existiert pro UTM-Anwendung ein Netz-Mainprozess (*utmnetm*). Zusätzlich wird für jede CMX-Kommunikationsbeziehung ein weiterer Netzprozess gestartet (*utmnetc*).

  Anbindung über die Socket-Schnittstelle (native TCP/IP)

  Für jede Socket-Listener-ID, die mit KDCDEF generiert wird (BCAMAPPL-Anweisung mit T-PROT=SOCKET), wird je ein Prozess (*utmnets*) gestartet.

  Das bedeutet:
  Für eine minimale Produktivanwendung („Ein-Prozess-Anwendung“) werden 16 Einträge gebraucht. Bei der Generierung von einem Schlüssel für die Semaphore kann folglich mindestens ein Dialogterminalprozess (*utmdtp*) parallel angeschlossen sein.

  Mit MAX...,SEMARRAY= kann ein Bereich von bis zu 1000 aufeinanderfolgenden Schlüsseln definiert werden. Mit der KDCDEF-Generierung mit MAX...,SEMKEY= können bis zu 10 einzelne Schlüssel für Semaphore definiert werden.

Filedeskriptoren

Ein Workprozess einer UTM-Anwendung belegt immer folgende Filedeskriptoren für:

- *stdout*
- *stdin*
- *stderr*
- die KDCFILE-Datei
- die SYSLOG-Datei
- das Shared Memory IPC
- das Shared Memory KAA
- das Shared Memory CACHE
- eine Named Pipe zum Mainprozess (*utmmain*)
- eine Named Pipe zum Loggingprozess (*utmlog*)

Weitere Filedeskriptoren werden benötigt, wenn die KDCFILE doppelt geführt wird, wenn PAGEPOOL-Dateien verwendet werden (MAX ...,PGPOOLFS=) oder wenn der Wiederanlaufbereich in mehrere Dateien aufgeteilt wird (MAX...,RECBUFFS=).

Ein Workprozess der Anwendung belegt kurzzeitig weitere Filedeskriptoren für:

- die aktuelle Benutzer-Protokolldatei (USLOG)
- Dump-Dateien bei Fehlern

Die Datei applifile

Die Datei wird bei der *open*UTM-Installation im Installationsverzeichnis eingerichtet und enthält die Namen aller Anwendungen, die im System seither gestartet wurden, inklusive Statusinformationen der Anwendungen. Außerdem enthält die Datei die Schlüssel (Keys) des Semaphors und Shared Memory Segments, die zur Kommunikation zwischen den externen Prozessen (Dialogterminal-, Drucker-, Timerprozess, lokale UPIC-Client-Programme) und den Workprozessen dienen. Die Schlüssel sind systemweit eindeutig zu vergeben.

ACHTUNG!

Die applifile ist eine UTM-interne Verwaltungsdatei. Sie dürfen sie **nicht** mit einem Editor öffnen. Die applifile könnte dadurch zerstört werden.

Zusätzliche Betriebsmittel für die Kommunikation über OSI TP

Falls OSI TP generiert ist, werden zusätzlich folgende Betriebsmittel belegt:

- ein XAPTP Shared Memory
- ein OSS Shared Memory
- ein Semaphor für den Zugriff auf das Shared Memory von OSS
- ein Semaphor für die globale Lockbörse von OSS
- pro Workprozess ein Semaphor für die OSS Taskbörse

– ein Filedeskriptor für das Shared Memory von OSS
– ein Filedeskriptor für das Shared Memory von XAPTP

3.4.2 Performance-Verbesserung: Größe der Datenbereiche im IPC-Shared-Memory anpassen

Den Shared-Memory-Bereich für die Inter-Prozess-Kommunikation (IPC-Shared-Memory-Bereich) benötigt die UTM-Anwendung für den Austausch der Nachrichten zwischen ihren Prozessen. Er wird von *open*UTM angelegt. Ist dieser Bereich zu klein, dann kann es zu Performance-Engpässen und zum Abbau von Verbindungen kommen.

Der größte Teil des IPC-Shared-Memory-Bereichs dient zur Ablage der Nachrichten, die zwischen den Prozessen einer Anwendung ausgetauscht werden. Dieser Bereich wird im folgenden Datenbereich genannt. Der restliche Teil dient zur Verwaltung der Prozesse und deren Verbindungen.

Der Datenbereich im IPC-Shared-Memory ist in Einheiten von 4 KB organisiert.

Die Größe des IPC-Shared-Memory wird von *open*UTM anwendungsspezifisch festgelegt. Sie ist im wesentlichen bestimmt durch die Anzahl der generierten Kommunikationspartner (MAX SEMARRAY) bzw. der generierten Semaphore (MAX SEMKEY). Siehe auch „Semaphore“ auf Seite 47.

Die Größe des Datenbereichs im IPC-Shared-Memory ist abhängig von der Art der Netzanbindung, die Sie für die UTM-Anwendung generiert haben:

- In Anwendungen mit multi-threaded Netzanbindung (Standard)
  (MAX NET-ACCESS = MULTI-THREADED)

  Pro generiertem Semaphorschlüssel legt *open*UTM auch hier einen Datenbereich von ca. 10 ∗ 4 KB an. Zusätzlich legt *open*UTM noch pro generiertem Kommunikationspartner einen Datenbereich von ca. 4 KB an.

- In Anwendungen mit single-threaded Netzanbindung
  (MAX NET-ACCESS = SINGLE-THREADED)

  Pro generiertem Semaphorschlüssel legt *open*UTM einen Datenbereich von ca. 10 ∗ 4 KB an.

Bei hohem Datenaufkommen auf den Verbindungen zu Kommunikationspartnern kann der von *open*UTM angelegte Datenbereich zu klein sein. Es kann dann zu Performance-Engpässen und somit zum Abbau von Verbindungen kommen. Um das zu vermeiden, können Sie die Größe des IPC-Shared-Memorys vergrößern. Dazu dienen die Umgebungsvariablen UTM_IPC_LETTER und UTM_IPC_EXTP_LETTER. Mit UTM_IPC_LETTER beeinflus-

sen Sie die absolute Größe des IPC-Shared-Memory, mit UTM_IPC_EXTP_LETTER den Datenbereich im IPC-Shared Memory, der einer einzelnen Verbindung maximal zur Verfügung steht.

Absolute Größe des Datenbereichs anpassen

Der Datenbereich im IPC-Shared-Memory wird unter den existierenden Verbindungen im Sinne von „first come - first serve“ aufgeteilt. Ist der gesamte Datenbereich belegt, werden Verbindungen abgebaut.

*open*UTM gibt dann folgende Meldung aus:

```
U189 IPC Engpass &IPCOBJ &IPCREAS
```

mit den Inserts &IPCOBJ=LETT und &IPCREAS=IPC FULL, EXTP FULL.

Um dies zu vermeiden, können Sie die absolute Größe des Datenbereichs mit Hilfe der Umgebungsvariablen UTM_IPC_LETTER vergrößern. In UTM_IPC_LETTER geben Sie die Anzahl der 4KB-Einheiten an, die das IPC-Shared Memory umfassen soll. Der kleinste erlaubte Wert ist 5 (entspricht 20KB).

Eine Änderung von UTM_IPC_LETTER wirkt sich erst nach dem nächsten Start der UTM-Anwendung aus. UTM_IPC_LETTER wird beim Anwendungsstart vom ersten Workprozess ausgewertet.

Bei sehr niedrigem Datenaufkommen können Sie andererseits mit UTM_IPC_LETTER den Datenbereich verkleinern. Damit können Sie den Overhead verringern, der durch die Verwaltung des IPC-Shared-Memorys durch das Betriebssystem benötigt wird.

3.4.2.1 Maximalen Datenbereich für das Nachrichtenaufkommen pro Verbindung anpassen

Grundsätzlich sind alle Verbindungen bei der Vergabe des Datenbereichs im IPC-Shared-Memory gleichberechtigt. Damit der Datenbereich aber nicht von einer Verbindung allein belegt wird, ist standardmäßig ein maximaler Bereich von 64 KB (16 ∗ 4KB) festgelegt, der von einer Verbindung zu einem Zeitpunkt belegt werden kann. Sind diese 64 KB von einer Verbindung bereits belegt, dann wird die Verbindung abgebaut.

*open*UTM gibt dann folgende Meldung aus:

```
U189 IPC Engpass &IPCOBJ &IPCREAS
```

mit den Inserts &&IPCOBJ=LETT und &IPCREAS=MAX ILETT oder MAX OLETT.

Um dies zu vermeiden, können Sie den Datenbereich, der im IPC-Shared Memory pro Verbindung maximal zur Verfügung steht, mit der Umgebungsvariablen UTM_IPC_EXTP_LETTER vergrößern.

Mit UTM_IPC_EXTP_LETTER können Sie die maximale Größe des Datenbereichs pro Verbindung in 4KB-Einheiten angeben. Als Standardwert ist 16 eingestellt. Der kleinste erlaubte Wert ist 1 (entspricht 4KB).

Eine Änderung von UTM_IPC_EXTP_LETTER wirkt sich erst nach dem nächsten Start der UTM-Anwendung aus. UTM_IPC_EXTP_LETTER wird beim Anwendungsstart vom ersten Workprozess ausgewertet.

4 UTM-Anwendung starten

Eine UTM-Anwendung wird gestartet, indem das Programm *utmmain* aufgerufen wird. Dieses Programm ist die Main-Funktion der Anwendung und läuft als Hintergrundprozess ab. *utmmain* erzeugt die Workprozesse (*utmwork*), den Timerprozess (*utmtimer*) und bei Bedarf Netzprozesse (*utmnet*, *utmnets, utnnetm*) und Druckerprozesse (*utmprint*, nur bei UNIX). Sehen Sie dazu auch das *open*UTM-Handbuch „Konzepte und Funktionen“.

Zu Testzwecken kann eine UTM-Anwendung auch mit einem Debugger gestartet werden, siehe Abschnitt „Starten einer UTM-Anwendung mit Debugger“ auf Seite 64.

4.1 Starten einer UTM-Anwendung in UNIX

Zum Starten einer Anwendung mit *utmmain* sind folgende Schritte notwendig:

1. Setzen Sie die Umgebungsvariable UTMPATH auf das *utm-dateiverzeichnis*. Es ist sinnvoll, wenn Sie dafür folgende Kommandos in die *.profile* bzw. in die *login-file* der Benutzerkennung aufnehmen, unter der die Anwendung abläuft:

   ```
   UTMPATH=utm-dateiverzeichnis
   export UTMPATH
   ```

   Näheres zum *utm-dateiverzeichnis* siehe Seite 186.

2. Erstellen Sie die Startparameterdatei wie ab Seite 58 beschrieben.

3. Starten Sie *utmmain* als Hintergrundprozess:

   utm-dateiverzeichnis`/ex/utmmain` *filebase* [*startparam-file*] `&`

   filebase
   : ist der vollqualifizierte Basisname für die UTM-Anwendung (Name des Dateiverzeichnisses der Anwendung).

   startparam-file
   : ist der vollqualifizierte Name der Datei, in der die Startparameter definiert sind. Wird dieser Parameter weggelassen, dann müssen die Startparameter in der Datei *filebase*`/startparameter` stehen.

utmmain erzeugt sowohl beim Start als auch während des Anwendungslaufs Meldungen auf *stdout*; Fehlermeldungen beim Start werden auf *stderr* ausgegeben, siehe auch Seite 68. Diese Meldungen können wie in folgendem Beispiel in eine Datei oder ein Filterprogramm umgelenkt werden.

Beispiel

Anwendung und Startparameterdatei befinden sich im Verzeichnis `/home/utmbsp`, die Startparameterdatei hat den Standardnamen `startparameter`.

Wen alle Ausgaben in Datei umgelenkt werden sollen, rufen Sie *utmmain* wie folgt auf:

```
utm-dateiverzeichnis/ex/utmmain /home/utmbsp 1>utmp.out  2>utmp.err &
```

Wenn die Ausgabe stattdessen von einem Programm oder Shellskript namens `filter` weiterbehandelt werden soll, rufen Sie *utmmain* wie folgt auf:

```
utm-dateiverzeichnis/ex/utmmain /home/utmbsp 2>&1 | filter
```

Mit dieser Methode kann z.B. auf Meldungen des Netzprozesses reagiert werden, was mit einem MSGTAC-Programm nicht möglich ist.

Netzprozesse

Beim Start erzeugt *utmmain* folgende Netzprozesse:

– Bei Multi-Threaded-Generierung (MAX NET-ACCESS=MULTI-THREADED) erzeugt *utmmain* einen oder mehrere *utmnet*-Prozesse.

– Bei Single-Threaded-Generierung (MAX NET-ACCESS=SINGLE-THREADED) wird ein *utmnetm*-Prozess erzeugt.

– Zusätzlich werden für die TCP/IP-Kommunikation einer oder mehrere Socket-Netzprozesse *utmnets* gestartet, siehe auch Abschnitt „Unterschiedliche Socket-Netzprozesse einsetzen“ auf Seite 188.

4.2 Starten einer UTM-Anwendung in Windows

Unter Windows können Sie eine UTM-Anwendung mit dem Programm *utmmain* oder als Dienst starten.

4.2.1 Starten mit utmmain

Zum Starten einer Anwendung mit *utmmain* sind folgende Schritte notwendig:

1. Ergänzen Sie für die Benutzerkennung, unter der die UTM-Anwendung laufen soll, die Variable PATH um *utm-dateiverzeichnis*`\ex`.
2. Erstellen Sie die Startparameterdatei wie ab Seite 58 beschrieben.
3. Öffnen Sie mit *Start - Programme - Eingabeaufforderung* ein Kommandofenster und geben Sie dort Folgendes ein:

   `utmmain` *filebase* [*startparam-file*]

   filebase
   : ist der vollqualifizierte Basisname für die UTM-Anwendung (Name des Dateiverzeichnisses der Anwendung). Sie können *filebase* auch als relativen Pfadnamen eingeben, z.B. als „.“ (Punkt), wenn Sie *utmmain* aus dem Verzeichnis *filebase* aufrufen.

   startparam-file
   : ist der vollqualifizierte Name der Datei, in der die Startparameter definiert sind. Wird dieser Parameter weggelassen, dann müssen die Startparameter in der Datei *filebase*`\startparameter` stehen.

   Für den Aufruf von *utmmain* Schritte können Sie auch ein Shortcut (= Kurzbefehl) erzeugen, so dass Sie die UTM-Anwendung per Mausklick oder Tastaturbefehl starten können, näheres siehe folgendes Beispiel.

utmmain erzeugt sowohl beim Start als auch während des Anwendungslaufs Meldungen auf *stdout*; Fehlermeldungen beim Start werden auf *stderr* ausgegeben, siehe auch Seite 68. Diese Meldungen können wie in folgendem Beispiel in eine Datei umgelenkt werden.

Beispiel

Die Anwendung befindet sich im Verzeichnis `C:\utmtest\beispiel`, die PATH-Variable Ihrer Benutzerkennung haben Sie um *utm-dateiverzeichnis*`\ex` ergänzt. Die Startparameterdatei besitzt den Standardnamen `startparameter` und befindet sich ebenfalls im Anwendungsverzeichnis.

Wenn Sie alle Meldungen in Datei umlenken möchten, öffnen Sie jetzt ein Kommandofenster und starten die UTM-Anwendung wie folgt:

```
cd C:\utmtest\beispiel
utmmain . 1>utmp.out  2>utmp.err
```

Wenn die Meldungen stattdessen von einem Programm oder Shellskript namens `filter` weiterbehandelt werden sollen, rufen Sie *utmmain* wie folgt auf:

```
utmmain . 2>&1 | filter
```

Mit dieser Methode kann z.B. auf Meldungen des Netzprozesses reagiert werden, was mit einem MSGTAC-Programm nicht möglich ist.

UTM-Anwendung über Shortcut starten

Für diesen Vorgang können Sie wie folgt ein Shortcut erzeugen, damit Sie die Anwendung per Mausklick oder über bestimmte Tastaturbefehle (hier `STRG+ALT+F3`) starten können:

1. Drücken Sie auf dem leeren Bildschirmhintergrund die rechte Maustaste, wählen Sie *Neu* und klicken Sie auf *Verknüpfung*. Das Fenster *Verknüpfung erstellen* wird geöffnet:
   - Geben Sie in das Feld *Befehlszeile* Folgendes ein:

     ```
     CMD /C utmmain . 1>utmp.out 2>utmp.err
     ```

     Durch `CMD` wird ein Kommandofenster geöffnet und die Befehle dort ausgeführt. `/C` heißt, dass das Fenster nach Beenden von *utmmain* geschlossen wird.
   - Klicken Sie auf *Weiter* und vergeben Sie im Fenster *Programmbezeichnung auswählen* einen sprechenden Namen, z.B. `start-utm`.
   - Klicken auf *Fertigstellen*. Das Fenster wird geschlossen, auf dem Bildschirm wird ein Symbol mit dem Namen `start-utm` angezeigt.
2. Klicken Sie mit der rechten Maustaste auf dieses Symbol und wählen Sie *Eigenschaften*. Klicken Sie das Registerblatt *Verknüpfung* an und führen dort folgende Schritte durch:
   - Tragen Sie in das Feld *Arbeitsverzeichnis* das Verzeichnis `C:\utmtest\beispiel` (= Anwendungsverzeichnis) ein. Damit sucht *utmmain* die Parameter in diesem Verzeichnis und legt die Dateien in diesem Verzeichnis ab.
   - Setzen Sie den Cursor in das Feld *Tastenkombination* und drücken Sie gleichzeitig `STRG`, `ALT` und `F3`. Wahlweise können Sie in diesem Fenster über den Button *Anderes Symbol...* auch ein anderes Symbol zuordnen.
   - Drücken Sie OK. Damit ist der Shortcut fertig, die Anwendung kann per Doppelklick oder `STRG+ALT+F3` gestartet werden.

4.2.2 Starten als Dienst

Eine UTM-Anwendung muss wie in Abschnitt „Anwendung als Dienst installieren“ auf Seite 32 beschrieben als Dienst installiert und konfiguriert werden. Dabei kann als Startart *Automatisch* eingestellt werden, so dass der Dienst bei jedem Hochfahren des Systems gestartet wird. Bei Startart *Manuell* muss der Dienst immer manuell gestartet werden.

Sie starten einen Dienst wie folgt (die Beschreibung gilt für Windows 2000, für andere Windows-Varianten gehen Sie analog vor):

1. Melden Sie sich unter einer Windows-Kennung an, die Administrationsberechtigung besitzt.
2. Rufen Sie die Systemsteuerung auf mit *Start - Einstellungen - Systemsteuerung*.
3. Klicken Sie auf *Verwaltung* und dann auf *Dienste*. Markieren Sie den gewünschten UTM-Dienst mit der rechten Maustaste; dieser hat immer den Namen `openUTM` *servicename*. *servicename* wird beim Installieren des Dienstes vergeben.
4. Wählen Sie im Kontextmenü auf den Befehl *Starten*. Der Dienst wird gestartet.

Beim Start und während des Anwendungslaufs erzeugt eine als Dienst gestartete Anwendung die gleichen Meldungen wie eine per *utmmain* gestartete Anwendung. Diese Meldungen werden immer in folgende Dateien geschrieben:

– Meldungen nach *stdout* in die Datei *filebase*`\utmp.out`
– Meldungen nach *stderr* in die Datei *filebase*`\utmp.err`, siehe auch Seite 68.

Wenn die Anwendung als Dienst gestartet und dabei das Systemkonto benutzt wird (Standardeinstellung), dann kann es vorkommen, dass einige Diagnose-Dateien im Systemverzeichnis abgelegt werden (z.B. in `C:\Win\system32`).

4.3 Startparameterdatei der Anwendung

Die Startparameterdatei wird vom Administrator der Anwendung mit einem beliebigen Editor erstellt.

In der Startparameterdatei werden aktuelle Ablaufparameter der Anwendung festgelegt. Dazu gehören z.B. die Anzahl der Workprozesse, mit der die Anwendung arbeiten soll, oder Parameter für einen angeschlossenen Resource Manager.

Die Startparameter können in einer oder mehreren Zeilen angegeben werden. Ein Präfix bestimmt, für wen die Startparameter bestimmt sind:

- Startparameter mit dem Präfix `.UTM` oder ohne Präfix werden von *open*UTM selbst interpretiert.
- Startparameter mit dem Präfix `.RMXA` leitet *open*UTM an den angeschlossenen Resource Manager (z.B. ein Datenbanksystem) zur Auswertung weiter (siehe Seite 76).

Die Reihenfolge der Startparameter für *open*UTM und das Datenbanksystem ist beliebig, das END-Kommando muss jedoch die Eingabe aller Startparameter abschließen.

Kommentare

Alle Zeilen mit einem „*“ (Stern) oder „#“ (Nummernzeichen) in Spalte 1 werden als Kommentar interpretiert. Kommentare können an beliebiger Stelle in der Startparameterdatei stehen. Damit können Sie z.B. einzelne Startparameter je nach Anwendungslauf aktivieren oder deaktivieren.

4.3.1 Startparameter für *open*UTM

Die Syntax der UTM-Startparameter ist im Folgenden dargestellt.

```
[.UTM] START   FILEBASE=filebase
               [ ,ASYNTASKS=number ]
               [ ,BTRACE={ ON | OFF } ]
               [ ,DUMP-CONTENT={ STANDARD | EXTENDED } ]
               [ ,DUMP-MESSAGE=event-code ]
               [ ,OTRACE={ ON | (SPI, INT, OSS, SERV, PROT) | OFF } ]
               [ ,STXIT={ ON | OFF } ]
               [ ,TASKS=number ]
               [ ,TASKS-IN-PGWT=number ]
               [ ,TESTMODE={ ON | OFF | FILE } ]
               [ ,UTM-MSG-DATE={ YES | NO } ]

[.UTM] END
```

In obiger Syntax werden die Parameter in einer Zeile **ohne Zeilenvorschub** angegeben und jeweils durch ein Komma getrennt.

Sie können die Parameter beim START-Kommando jedoch auch auf mehrere Zeilen verteilen. In diesem Fall müssen Sie in jeder Zeile das Kommando START vor den Parameter stellen.

Syntax-Prüfung beim Start der Anwendung

– Erkennt *open*UTM bei der Überprüfung der Startparameter einen Syntaxfehler, dann gibt *open*UTM die Meldung K038 aus, setzt für den betroffenen Startparameter den Standardwert, sofern vorhanden, und startet die Anwendung.

– Bei einem Syntaxfehler im Parameter FILEBASE kann die Anwendung **nicht** gestartet werden, da kein Standardwert für diesen Parameter existiert.

Bedeutung der Kommandos

START Mit diesem Kommando gibt man die für den Lauf einer UTM-Anwendung erforderlichen UTM-Startparameter an. Die Anwendung wird nach der Eingabe aller Startparameter sofort gestartet.

END Dieses Kommando schließt die Eingabe der Startparameter ab.

Bedeutung der Operanden

FILEBASE=filebase

Basisname für die KDCFILE und die Benutzer-Protokolldatei, wie er bei der KDCDEF-Generierung mit MAX...,KDCFILE= angegeben wurde. Bei Angabe eines ungültigen Namens wird der Start der Anwendung abgebrochen.

ASYNTASKS=number

Anzahl der Workprozesse, die max. für asynchrone Vorgänge arbeiten sollen.

Standard: in MAX...,ASYNTASKS=*number* festgelegte Anzahl
Minimalwert: 0
Maximalwert: in MAX...,ASYNTASKS=*number* festgelegte Anzahl

BTRACE= Ein-/Ausschalten der BCAM-Tracefunktion.

ON Beim Start der Anwendung wird die BCAM-Tracefunktion eingeschaltet. Es werden alle verbindungsspezifischen Ereignisse in der BCAM-Tracedatei aufgezeichnet. Sehen Sie dazu auch das *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

OFF Beim Start der Anwendung bleibt die BCAM-Tracefunktion ausgeschaltet.

Standard: OFF

DUMP-CONTENT=

gibt an, ob *open*UTM die prozessübergreifenden Speicherbereiche in allen Dumps einer Dump-Dateigeneration, d.h. für alle Prozesse, abzieht oder nur im Dump des Prozesses, der den Anwendungsabbruch verursacht hat.

STANDAR**D**

Wenn *open*UTM eine Dump-Dateigeneration erzeugt, dann sind prozessübergreifende Speicherbereiche nur im Dump des ersten Prozesses (Verursacher) enthalten. Für die Diagnose ist das normalerweise ausreichend.

Standard: STANDARD

EXTENDED

Die prozessübergreifenden Speicherbereiche sind in allen Dumps einer DUMP-Dateigeneration enthalten.

Diesen Wert sollten Sie nur auf besondere Anforderung des Service einstellen.

DUMP-MESSAGE= *event-code*

event-code gibt das Ereignis an, bei dem UTM bei eingeschaltetem Testmodus einen UTM-Dump mit Kennzeichen MSGDMP erzeugt. Ein Dump wird nur von der Task erstellt, in der das Ereignis eingetreten ist; die Anwendung wird dabei nicht beendet.

Der Wert von DUMP-MESSAGE kann per Administration zurückgesetzt werden, z.B. durch WinAdmin oder das Kommando KDCDIAG DUMP-MESSAGE=K000.

Sie können für *event-code* Folgendes angeben:

– *event-code*=K*nnn* (K-Meldung)

  Der UTM-Dump wird erzeugt, wenn die Meldung K*nnn* auftritt.
  Bei den Meldungsnummern K023, K043, K061, K062 wird nur einmal ein Dump erzeugt, danach wird *event-code* automatisch zurückgesetzt. Bei allen anderen Meldungen wird solange bei jedem Auftreten der Meldungsnummer ein Dump erzeugt, bis der Wert per Administration (s.o.) zurückgesetzt wird.

– *event-code*=CC*nn* (Kompatibler KDCS-Returncode)

  Der UTM-Dump wird erzeugt, wenn in einem Teilprogramm ein KDCS-Returncode der Form *nn*Z auftritt, z.B 14Z. Danach wird der Wert zurückgesetzt.

– *event-code*=K*xyz* (Inkompatibler KDCS-Returncode)

  Der UTM-Dump wird erzeugt, wenn in einem Teilprogramm der inkompatible KDCS-Returncode K*xyz* auftritt, z.B KD10. Danach wird der Wert zurückgesetzt.

– *event-code*=S*xyy* (SIGN-Returncode)

  Der UTM-Dump wird erzeugt, wenn beim SIGN-Aufruf ein Returncode der Form *xyy* auftritt. Hierbei sind *x* und *yy* die Werte in KCRSIGN1 bzw. KCRSIGN2, wobei KCRSIGN1 den Wert U, I, A oder R haben muss. SU13 bedeutet z.B. „neues Passwort zu kurz“. Der Dump wird unabhängig davon erzeugt, ob in der Anwendung ein Anmeldevorgang generiert ist oder nicht. Anschließend wird der Wert zurückgesetzt.

Hinweis
Bei allen KDCS-Returncodes ≥70Z und den zugehörigen inkompatiblen KDCS-Returncodes, bei denen kein PENDER-Dump geschrieben wird (z.B. 70Z/K316), wird auch kein DUMP mit Kennzeichen MSGDMP erzeugt.

OTRACE= Ein-/Ausschalten der Tracefunktion von OSS.
Der OSS-Trace wird zur Diagnose bei Problemen mit OSI-TP-Verbindungen der Anwendung benötigt. Sehen Sie dazu auch das *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“ und das Handbuch „OSS (BS2000/OSD)“.

ON Schaltet die OSS-Tracefunktion ein.
Es werden die Trace-Records der Typen SPI, INT, OSS, SERV und PROT protokolliert. Beim Einschalten der OSS-Tracefunktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei.

(SPI, INT, OSS, SERV, PROT)
Schaltet die OSS-Tracefunktion ein. Es werden die Trace-Records des angegebenen Typs protokolliert. Die Reihenfolge in der die Trace-Records angegeben werden ist beliebig.

SPI
Das XAP-TP System Programming Interface wird protokolliert.

INT
Der interne Ablauf im XAP-TP-Baustein wird protokolliert.

OSS
Die OSS-Aufrufe werden protokolliert.

SERV
Die OSS-internen Trace Records vom Typ O_TR_SERV werden protokolliert.

PROT
Die OSS-internen Trace Records vom Typ O_TR_PROT werden protokolliert.

OFF Schaltet die OSS-Tracefunktion aus.

STXIT= Signalbehandlung bei *open*UTM aktivieren.

ON Signalbehandlung bei *open*UTM ist eingeschaltet.

Standard: ON

OFF Die Standard-Fehlerbehandlung für Signale wird ausgeschaltet.

STXIT=OFF bewirkt in UNIX-Systemen, dass bei jedem Beenden eines Workprozesses zusätzliche Diagnose-Speicherauszüge (gcore-Dumps) im Basis-Dateiverzeichnis der Anwendung erzeugt werden. Die gcore-Dumps werden nur geschrieben, wenn im Dateiverzeichnis `/bin` das Programm *gcore* zur Verfügung steht. Die gcore-Dumps werden durch eine K078 Meldung angekündigt:
`K078 STXIT OFF in utmwork: termination of utmwork process creates gcore-dump`

TASKS=number
Anzahl der Workprozesse, die für diese Anwendung arbeiten sollen.

Standard: in MAX...,TASKS=*number* festgelegte Anzahl
Minimalwert: 1
Maximalwert: in MAX...,TASKS=*number* festgelegte Anzahl

TASKS-IN-PGWT=number
Maximale Anzahl der Prozesse, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen wie z.B. der KDCS-Aufruf PGWT ablaufen dürfen (Operand PGWT= in den KDCDEF-Anweisungen TAC und TACCLASS).

Standard: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.
Minimalwert: 1 falls MAX...,TASKS-IN-PGWT > 0 generiert wurde, sonst 0.
Maximalwert: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.

TESTMODE= Testmodus einschalten.
Sehen Sie hierzu auch *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“, Fehlerdiagnose.

ON Beim Start der Anwendung soll der Testmodus eingeschaltet werden. Im Testmodus werden zusätzliche UTM-interne Plausibilitätsprüfungen bei internen Prozeduraufrufen durchgeführt und interne Trace-Informationen sowohl im KTA als auch im XAP-TP-Baustein bei OSI-TP-Anwendungen aufgezeichnet. Der Testmodus sollte nur zur Diagnose von UTM-Fehlern auf Empfehlung des Systemberaters eingeschaltet werden.

Mit MAX...,IPCTRACE= (siehe *open*UTM-Handbuch „Anwendungen generieren“, MAX-Anweisung) können Sie bei der KDCDEF-Generierung die Anzahl Trace-Informations-Einträge angeben, die bei TESTMODE=ON geschrieben werden. IPCTRACE sollte nur zur Diagnose von schwerwiegenden UTM-Fehlern auf Empfehlung des Systemberaters versorgt werden.

OFF Beim Start der Anwendung soll der Testmodus ausgeschaltet bleiben.

Standard: OFF

FILE Beim Start der Anwendung wird der Testmodus eingeschaltet. Zusätzlich werden bei jedem Überlauf der KTA-Trace-Area die Diagnosedaten auf Datei geschrieben, um einen evtl. Diagnosedatenverlust zu vermeiden.

Der Dateiname setzt sich zusammen aus dem Basisnamen *filebase* und der PID des jeweiligen Workprozesses, d.h. es wird bei einer UTM-Produktivanwendung pro Workprozess folgende Datei angelegt:

filebase.KTATRC.*pid* (*pid* max. 4-stellig)

UTM-MSG-DATE=

gibt an, ob UTM-Meldungen mit oder ohne Datum/Uhrzeit auf *stdout* und *stderr* ausgegeben werden.

YES Alle UTM-Meldungen auf *stdout* und *stderr* werden mit dem Datum/Uhrzeit-Präfix ausgezeichnet.

Standard: YES

NO Die UTM-Meldungen erhalten kein Datum/Uhrzeit-Präfix.

Der mit UTM-MSG-DATE gesetzte Wert kann mit der Umgebungsvariablen UTM_MSG_DATE geändert werden.

Beispiel für den Inhalt einer Startparameterdatei

UNIX

```
START FILEBASE=/home/utmbsp
START TASKS=2
END
```

Windows

```
START FILEBASE=C:\utmtest\beispiel
START TASKS=2
END
```

4.4 Starten einer UTM-Anwendung mit Debugger

Um den Mehrprozess-Betrieb für Ihre Anwendung zu testen, können Sie Ihre Anwendung unter der Kontrolle einer Testhilfe ablaufen lassen. Dazu starten Sie den Mainprozess aus der Shell im Testmodus; die Workprozesse werden anschließend unter der Kontrolle der Testhilfe gestartet. Bei diesem Test kann der volle Funktionsumfang von *open*UTM auch bei verteilten Anwendungen getestet werden. Insbesondere werden wie bei einer „echten“ Produktionsanwendung Wartezeiten mit dem Timerprozess überwacht.

4.4.1 Starten mit Debugger in UNIX

In UNIX können Sie Testhilfen wie *dbx*, *xdb*, *debug*, *gdb*, *sdb*, *adb*, COBOL-Testhilfen usw. einsetzen.

Für den Start der Anwendung sind folgende Arbeitsschritte nötig:

1. Sie starten den Mainprozess als Hintergrundprozess wie bei einer UTM-Produktivanwendung. Sie müssen jedoch zusätzlich den Parameter TEST angeben.

   utm-dateiverzeichnis`/ex/utmmain` *filebase startparam-file* `TEST &`

   Die Startparameterdatei müssen Sie immer vollqualifiziert angeben, auch dann, wenn der Standardname verwendet wird.

   Der Mainprozess *utmmain* erzeugt dann nur den Timerprozess und weitere Betriebsmittel wie z.B. die Pipe zur Kommunikation mit den Workprozessen. Es werden aber keine Workprozesse erzeugt.

2. Der Mainprozess fordert Sie mit der Meldung U244 auf, einen Workprozess zu starten. Den ersten Workprozess *utmwork* müssen Sie unter der Kontrolle der Testhilfe mit folgenden Argumenten starten:

   utmversion appliname filebase startparam-file pid `Y`

   Die gesamte Anweisung muss **ohne Zeilenvorschub** geschrieben werden, auch wenn sie mehr als eine Zeile beansprucht. Die Argumente haben folgende Bedeutung:

utmversion	Verwendete *open*UTM(UNIX)-Version, z.B. V05.2A00
appliname	Name der Anwendung wie in MAX APPLINAME generiert.
filebase	Dateiverzeichnis der Anwendung (vollqualifiziert)
startparam-file	Datei, in der die Startparameter definiert sind (vollqualifiziert)
pid	Prozess-ID des Vorgängerprozesses, hier geben Sie normalerweise den Wert 0 an
Y	Kennzeichen für ersten Workprozess

Beispiel für dbx

```
run V05.2A00 sample /home/utmbsp /home/utmbsp/startparameter 0 0 Y
```

3. Danach können Sie die weiteren Workprozesse starten. Dazu geben Sie *utmwork* mit den gleichen Argumenten ein. Sie müssen nur als Kennzeichen des Workprozesses statt Y den Wert N angeben.

   utmversion appliname filebase startparam-file pid wid `N`

   Beispiel für dbx

   ```
   run V05.2A00 sample /home/utmbsp /home/utmbsp/startparameter 0 0 N
   ```

   Sie können nicht mehr Workprozesse starten als in der Startparameterdatei bei TASKS= angegeben wurden. Die Maximalzahl der Workprozesse wird bei der Generierung in der MAX-Anweisung festgelegt. Versuchen Sie mehr Workprozesse zu starten, führt dies zu Startfehlern.

4. Nach dem Beenden der UTM-Anwendung ist der Mainprozess mit dem Kommando `kill` zu löschen

   Beispiel

   ```
   kill -9 1234
   ```

   Dabei ist `1234` die Prozess-ID von *utmmain*.

Verhalten bei Fehlern

Falls ein Workprozess einen PEND ER-Dump erzeugt, wird dieser Workprozess beendet. Der Mainprozess startet dann keinen neuen Workprozess, sondern fordert Sie mit der Meldung U244 auf, den Workprozess neu zu starten. Sie müssen den Workprozess dann mit dem Kennzeichen E starten. Als *pid* geben Sie die Prozess-ID des Vorgänger-Workprozesses an, der sich mit PEND ER beendet hat.

utmversion appliname filebase startparam-file pid `E`

Wird ein Workprozess fehlerhaft beendet, sollten Sie vor dem nächsten Start der Anwendung das Tool KDCREM aufrufen. Sehen Sie dazu Abschnitt „Das Tool KDCREM“ auf Seite 72.

Beenden der Anwendung

Die Anwendung beenden Sie wie sonst auch durch die UTM-Administration oder mit dem Tool KDCSHUT. Sehen Sie dazu das Kapitel „UTM-Anwendung beenden“ auf Seite 69. Falls sich die Anwendung nicht normal beendet, sollten Sie das Tool KDCREM aufrufen. Sehen Sie dazu Abschnitt „Das Tool KDCREM“ auf Seite 72.

4.4.2 Starten mit Debugger in Windows

In Windows verwenden Sie den Debugger, der im Microsoft Visual C++ Developer Studio integriert ist.

Für den Start der Anwendung sind folgende Arbeitsschritte nötig:

1. Sie starten den Mainprozess wie bei einer UTM-Produktivanwendung. Sie müssen jedoch zusätzlich den Parameter TEST angeben.

   Öffnen Sie dazu ein Fenster *Eingabeaufforderung*, wechseln Sie in das Anwendungsverzeichnis und geben Sie Folgendes ein:

   ```
   utmmain . startparam-file TEST 1>utmp-out.txt <nul
   ```

   Dabei muss die PATH-Variable entsprechend gesetzt sein. Die Ausgabe auf *stdout* wird hier in Datei umgelenkt. `nul` entspricht `/dev/null` in UNIX.

   Der Mainprozess *utmmain* erzeugt dann nur den Timerprozess und weitere Betriebsmittel wie z.B. die Pipe zur Kommunikation mit den Workprozessen. Es werden aber keine Workprozesse erzeugt.

   Der Mainprozess fordert Sie mit der Meldung U244 auf, einen Workprozess zu starten. U244 kann z.B. so aussehen:

   ```
   U244 utmmain: Bitte ./utmwork starten mit Argumenten:
   V05.2A00 sample01 . ./startp.std 0 0 Y
   ```

   Die zweite Zeile benötigen Sie später für den Debugger. Das Argument `0` bedeutet, dass dieser Prozess neu gestartet wird, d.h. er ersetzt keinen zuvor beendeten Workprozess. `Y` bedeutet zusätzlich, dass dies der erste Workprozess ist.

   Sie dürfen das Fenster jetzt nicht schließen, sonst wird der Mainprozess beendet!

2. Den ersten Workprozess *utmwork* müssen Sie unter der Kontrolle des Visual C++-Debuggers starten. Dazu öffnen Sie Ihr Anwendungsprojekt mit dem Developer Studio, z.B. indem Sie im Explorer auf den Projektarbeitsbereich (*projektname*`.dsw`) doppelklicken. Führen Sie folgende Schritte durch:

   a) Klicken Sie in der Menüsleiste *Projekt - Einstellungen*; damit wird das Fenster *Projekteinstellungen* geöffnet.

      – Wählen Sie bei *Einstellungen für* den Wert *Win32 Debug* aus.

      – Klicken Sie das Blatt *Debug* an und tragen Sie in das Feld *Programmargumente* die Argumente aus der Meldung U244 ein, im Beispiel `V05.2A00 sample01 . ./startp.std 0 0 Y`.

        Diese Argumente haben immer das Format

        utmversion appliname filebase startparam-file pid `Y`

Dabei ist *utmversion* die Version von *open*UTM, *appliname* der Name der Anwendung wie in MAX APPLINAME generiert, *pid* die Prozess-ID des Vorgänger-Workprozesses (hier 0) und `Y` das Kennzeichen für den ersten Workprozess.

– Klicken Sie OK. Damit kann der erste Workprozess gestartet werden.

b) Starten Sie den Debugger *Erstellen - Debug starten - In Aufruf springen*. Der Workprozess wird gestartet und das Fenster *Quellcode suchen* geöffnet.

Klicken Sie dort auf *Abbrechen*. Der Debugger springt an den Programmanfang, Sie können die gewünschten Debug-Funktionen ausführen.

3. Danach können Sie die weiteren Workprozesse starten.

Dazu wiederholen Sie die Schritte 2. a) und b), nur dass Sie bei *Programmargumente* für das Kennzeichen des Workprozesses den Wert `N` statt `Y` angeben:

utmversion appliname filebase startparam-file pid wid `N`

Sie können nicht mehr Workprozesse starten als in der Startparameterdatei bei TASKS= angegeben wurden. Die Maximalzahl der Workprozesse wird bei der Generierung in der MAX-Anweisung festgelegt. Versuchen Sie mehr Workprozesse zu starten, führt dies zu Startfehlern.

4. Nach dem Beenden der Workprozesse müssen Sie den Mainprozess explizit beenden, indem Sie das zugehörige Fenster schließen oder den Prozess *utmmain.exe* mit dem Task-Manager beenden.

4.5 Kaltstart und Warmstart

Man versteht bei *open*UTM unter:

- Kaltstart:
  Starten nach normaler Beendigung der UTM-Anwendung oder nach Neugenerierung.
- Warmstart:
  Starten nach abnormaler Beendigung der UTM-Anwendung.

Kaltstart bei *open*UTM

Vor dem ersten Start einer Anwendung haben Sie die KDCFILE mit dem Generierungstool KDCDEF eingerichtet. Wenn die UTM-Anwendung normal beendet wurde, greift *open*UTM bei einem erneuten Start der Anwendung wieder auf dieselbe KDCFILE zu und führt einen Kaltstart durch. Nach erfolgreichem Start meldet *open*UTM:

```
K051 Kaltstart für Anwendung appliname mit UTM V05.2A00 erfolgreich
```

Warmstart bei *open*UTM

Bei einem Wiederanlauf bringt *open*UTM die KDCFILE in einen konsistenten Zustand. Nach erfolgreichem Start meldet *open*UTM:

```
K050 Warmstart für Anwendung appliname mit UTM V05.2A00 erfolgreich.
```

Dabei ist zu beachten, dass sich UTM-S und UTM-F im Umfang der Wiederanlauf-Funktionen unterscheiden. Sehen Sie dazu auch im *open*UTM-Handbuch „Konzepte und Funktionen“, Abschnitt „Fehlertoleranz und Wiederanlauf“.

Hat sich eine UTM-Datenbank-Anwendung abnormal beendet (Systemabbruch oder UTM-Anwendungsabbruch), dann muss der Administrator des Datenbanksystems die Datenbank vor dem Warmstart in einen ordnungsgemäßen Zustand bringen. Beim Warmstart der UTM-Datenbank-Anwendung führt *open*UTM eine gemeinsame Recovery Phase durch.

4.6 Fehlermeldungen beim Start

Wird der Start einer UTM-Anwendung oder eines Prozesses wegen eines Fehlers abgebrochen, gibt *open*UTM in der Regel die Meldungen K049 und/oder K078 aus. Die Meldung K078 kann in mehreren Varianten auftreten. Die Bedeutung dieser Meldungen und deren Returncodes sind ausführlich im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“ erklärt.

Startfehler können beim Start eines jeden Workprozesses auftreten.

5 UTM-Anwendung beenden

Eine UTM-Anwendung kann

– normal beendet werden durch Administrationsaktionen oder durch das Tool KDCSHUT
– abnormal beendet werden durch Fehler oder per Administration

Nach einem Anwendungsende sind evtl. noch systemglobale Betriebsmittel zu löschen, bevor die Anwendung erneut gestartet wird.
Sehen Sie dazu Abschnitt „Das Tool KDCREM“ auf Seite 72.

5.1 UTM-Anwendung normal beenden

Eine UTM-Anwendung beendet der UTM-Administrator normal, indem er z.B. folgendes UTM-Administrationskommando an einem Administrationsterminal eingibt:

```
KDCSHUT NORMAL
```

oder

```
KDCSHUT WARN,TIME=time
```

In Anwendungen, die verteilte Transaktionsverarbeitung verwenden, sollte eine Anwendung immer mit KDCSHUT WARN beendet werden, denn dies erlaubt eine geordnete Beendigung der offenen verteilten Transaktionen.

Nach Eingabe des Kommandos führt *open*UTM die folgenden Aktionen durch:

– Es werden alle Aufträge abgearbeitet, die noch in der UTM-Warteschlange sind.
– Die Verbindungen zu Terminals werden abgebaut.
– KDCFILE, System-Protokolldatei und Benutzer-Protokolldatei werden in einen konsistenten Zustand gebracht und ordnungsgemäß geschlossen.
– Alle Prozesse der Anwendung werden beendet.

Um eine UTM-Anwendung normal zu beenden, können Sie an Stelle des Kommandos KDCSHUT auch die entsprechende Funktion bei WinAdmin oder an der Administrations-Programmschnittstelle verwenden.

5.2 Das Tool KDCSHUT - UTM-Anwendung auf Shell-Ebene normal beenden

Das Tool KDCSHUT ist eine einfache Methode, die Anwendung zu beenden, ohne sich als Administrator bei der Anwendung anmelden zu müssen. Das Tool KDCSHUT wirkt wie das Administrationskommando KDCSHUT N bzw. das Kommando KDCSHUT W, TIME=.

Das Tool KDCSHUT wird wie folgt aufgerufen:

UNIX: *utm-dateiverzeichnis*`/ex/kdcshut␣`*filebase*`␣`[*time*]

Windows: *utm-dateiverzeichnis*`\ex\kdcshut␣`*filebase*`␣`[*time*]

filebase ist der Basisname der Anwendung;
time ist die Wartezeit in Minuten, bis sich die Anwendung beendet.

5.3 Dienst beenden in Windows

Wenn eine UTM-Anwendung als Dienst gestartet wurde, dann kann sie entweder so beendet werden, als ob sie nicht als Dienst gestartet wurde, oder sie kann als Dienst beendet werden. Dazu gehen Sie ähnlich wie beim Starten vor:

1. Melden Sie sich unter einer Windows-Kennung an, die Administrationsberechtigung besitzt.
2. Rufen Sie die Systemsteuerung auf mit *Start - Einstellungen - Systemsteuerung*.
3. Klicken Sie auf *Dienste* und markieren Sie den gewünschten UTM-Dienst; dieser hat immer den Namen `openUTM`*servicename*.
4. Drücken den Button *Beenden*; der Dienst und damit die Anwendung werden normal beendet.

Wird der Windows-Rechner heruntergefahren, dann werden der Dienst und damit die Anwendung ebenfalls normal beendet.

5.4 UTM-Anwendung abnormal beenden

Eine UTM-Anwendung kann durch folgende Ereignisse abnormal beendet werden:

– UTM-interner Fehler
– Fehler in der Systemumgebung und Shutdown des UNIX-Systems
– UTM-Administrationskommando KDCSHUT KILL (oder durch die entsprechende Funktion bei WinAdmin oder KDCADMI)
– Anwenderfehler

Bei abnormaler Beendigung werden folgende Aktionen ausgeführt:

- Alle Transaktionen, die gerade von den einzelnen Workprozessen bearbeitet werden, werden abgebrochen.
- Die Verbindungen zu allen Kommunikationspartnern der Anwendung werden abgebaut.
- Für jeden Workprozess der Anwendung wird ein UTM-spezifischer Speicherauszug (UTM-Dump) erzeugt. Siehe dazu auch *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.
- Alle Prozesse der Anwendung werden beendet und alle Dateien werden geschlossen. Es wird nicht versucht, die KDCFILE in einen konsistenten Zustand zu bringen. Dies geschieht erst bei einem erneuten Start der Anwendung.

Nach einem abnormalen Beenden der Anwendung sollten Sie als erstes die Ursache für den Abbruch feststellen. Dazu suchen Sie im Protokoll der Workprozesse auf *stdout* die Meldung K060. Diese Meldung enthält als Insert den Dump-Fehlercode, der genaue Auskunft über die Abbruch-Ursache gibt. Die Dump-Ursache können Sie auch als Teil des Dateinamens der UTM-Dumps finden. Die Bedeutung des Dump-Fehlercodes ist im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“ bei der Meldung K060 beschrieben. Dabei gibt es drei Möglichkeiten:

- Der Dump-Fehlercode sagt aus, dass ein KDCDEF-Operand verändert werden muss. Dann ist die KDCFILE neu zu generieren. Wenn Sie die Benutzerdaten im Pagepool erhalten wollen, gehen Sie wie folgt vor:
  – Warmstart mit ASYNTASKS=0, TASKS=1
  – Anwendung normal beenden mit KDCSHUT NORMAL
  – alte KDCFILE sichern
  – neue KDCDEF-Generierung mit dem veränderten Operanden
  – Übertragung der Benutzerdaten mit KDCUPD aus der alten in die neue KDCFILE
  – Start der Anwendung mit der neuen aktualisierten KDCFILE

- Der Dump-Fehlercode nennt als Ursache
  - Fehler vom Datenverwaltungssystem
  - Hardware-Fehler eines Terminals

  Ist der Fehler behoben, können Sie die Anwendung erneut starten, *open*UTM führt dann automatisch einen Warmstart durch.
- Es liegt ein Systemfehler vor. In diesem Fall erstellen Sie Diagnoseunterlagen und schreiben eine Fehlermeldung an den Systembetreuer. Dazu bereiten Sie die UTM-Dumps aller Workprozesse der Anwendung mit dem Tool KDCDUMP auf. Weitere Unterlagen sind die *stdout*- und *stderr*-Protokolle, die *gcores* (bei UNIX), das *utmwork*-Programm, die Symboltabelle *nmutmwork* (bei UNIX), die KDCDEF-Steueranweisungen und eine Auswertung der System-Protokolldatei.

  Ein Warmstart mit derselben KDCFILE ist in diesem Fall nicht immer erfolgreich. Kann kein Warmstart durchgeführt werden, müssen Sie die KDCFILE mit KDCDEF neu generieren.

Bei einer abnormalen Beendigung ist vor dem Neustart der Anwendung das Tool KDCREM aufzurufen (siehe folgenden Abschnitt).

5.5 Das Tool KDCREM

Mit dem Tool KDCREM werden nach einem Anwendungsende evtl. noch vorhandene Semaphore und Shared Memories sowie Statusinformationen der Anwendung, die in der Datei `applifile` im *utm-dateiverzeichnis* enthalten sind, gelöscht bzw. zurückgesetzt. Sehen Sie dazu auch den Abschnitt „Systemglobale Betriebsmittel einer Anwendung“ auf Seite 47.

ACHTUNG!

Nach der abnormalen Beendigung des Mainprozesses einer UTM-Produktivanwendung (z.B. durch einen UNIX- bzw. Windows-Systemfehler, System-Shutdown oder das Signal SIGKILL) **muss** KDCREM vor einem Anwendungsstart aufgerufen werden.

Aufruf von KDCREM

UNIX: *utm-dateiverzeichnis*`/ex/kdcrem␣`*filebase*

Windows: *utm-dateiverzeichnis*`\ex\kdcrem␣`*filebase*

filebase ist der Basisname der Anwendung, deren Semaphore, Shared Memories und Statusinformationen in der Datei `applifile` in *utm-dateiverzeichnis* gelöscht werden sollen.

6 UTM-Datenbank-Anwendung

Dieses Kapitel gibt einen zusammenhängenden Überblick über das, was für den Einsatz von Datenbanken (= Resource Managern) unter *open*UTM zu beachten ist. *open*UTM verwendet für die Kopplung die von X/Open genormte XA-Schnittstelle.

*open*UTM (UNIX) unterstützt die Koordination mit folgenden Datenbank-Systemen:

– INFORMIX

– ORACLE

– alle anderen Datenbank-, Datei- und Queuing-Systeme, die die XA-Schnittstelle (X/Open-Standard) unterstützen, z.B.: SYBASE, ADABAS, DB2, O2 ,SQL-Server, ISAM/XA, MQSeries, ...

*open*UTM (Windows) unterstützt die Koordination mit folgenden Datenbank-Systemen:

– INFORMIX

– ORACLE

– Microsoft SQL Server

Nähere Information über das Konzept der koordinierten Zusammenarbeit finden Sie im *open*UTM-Handbuch „Konzepte und Funktionen“.

6.1 UTM-Datenbank-Anschluss generieren

Sie müssen den UTM-Datenbank-Anschluss in der KDCDEF-Anweisung RMXA generieren. Dort geben Sie an:

– den Namen der *xa_switch_t*-Struktur wie er von der eingesetzten Datenbank vorgegeben ist.

– ob die CAE-Spezifikation benutzt werden soll

– Bei Windows: ob die *xa_switch_t*-Struktur mit dllimport adressiert wird. Bei der Kopplung mit ORACLE und MQSeries muss immer über dllimport adressiert werden.

Näheres finden Sie im *open*UTM-Handbuch „Anwendungen generieren“ bei der Beschreibung der RMXA-Anweisung.

6.2 UTM-Datenbank-Anwendung binden in UNIX

Für die UTM-Datenbankkopplung müssen Sie zusätzliche Module in den UTM-Workprozess einbinden. Im Folgenden sind diese Module für die einzelnen Datenbanksysteme aufgelistet. Die Richtigkeit der Namen von Modulen, die Bestandteil des jeweiligen Datenbanksystems sind, sollten Sie anhand der Benutzerhandbücher zu diesem Datenbanksystem überprüfen.

Sie können sich die Arbeit erleichtern, indem Sie die Beispielanwendung verwenden, die mit *open*UTM ausgeliefert wird, siehe auch Seite 210. Mit Hilfe dieser Beispielanwendung lässt sich auf einfache Art und Weise eine UTM-Datenbank-Anwendung erstellen, die alle jeweils benötigten Datenbankbibliotheken enthält. Diese Datenbank-Anwendung kann als Vorlage für Ihre eigenen Anwendung dienen; Sie können z.B. das dabei erzeugte Makefile anpassen, siehe Abschnitt „Binden mit Makefile“ auf Seite 20.

Kopplung mit INFORMIX

In der folgenden Tabelle sind alle UTM-Module enthalten, die für die Datenbankkopplung in den UTM-Workprozess eingebunden werden müssen.

Modul	aus Bibliothek (ohne shared libraries)	aus Bibliothek (mit shared libraries)
`kcxxacae.o` `kcxmsgs.o`	*utm-dvz*`/sys/libwork.a` *utm-dvz*`/sys/libwork.a`	*utm-dvz*`/libutm.a` *utm-dvz*`/libutm.a`

Mit dem Kommando `esql -libs` können Sie sich die erforderlichen Bibliotheken ausgeben lassen.

Kopplung mit ORACLE 9i

In der folgenden Tabelle sind alle UTM-Module enthalten, die für die Datenbankkopplung in den UTM-Workprozess eingebunden werden müssen.

Modul	aus Bibliothek (ohne shared libraries)	aus Bibliothek (mit shared libraries)
`kcxxacae.o` `kcxmsgs.o`	*utm-dvz*`/sys/libwork.a` *utm-dvz*`/sys/libwork.a`	*utm-dvz*`/libutm.a` *utm-dvz*`/libutm.a`

Zusätzlich müssen eine Reihe von ORACLE-Modulen eingebunden werden. Welche dies sind, können Sie mit Hilfe der mit ORACLE ausgelieferten Beispielprogramme und -Prozeduren ermitteln, z.B. mit folgendem Aufruf:

ORACLE 9i: *ORACLE-dvz*`/precomp/demo/proc/make -f demo_proc.mk sample1`

ORACLE-dvz bezeichnet das Installationsverzeichnis der betreffenden ORACLE-Version. Bei 64-bit-ORACLE-Plattformen müssen Sie `demo_proc32.mk oder demo_proc64.mk` angeben, abhängig davon, ob eine 64-bit-Anwendung erstellt werden soll.

Das Präprozessor Flag `release_cursor=yes` ist unbedingt zu setzen. Siehe dazu das Benutzerhandbuch zu ORACLE.

6.3 UTM-Datenbank-Anwendung binden in Windows

In Windows erstellen Sie eine UTM-Datenbank-Anwendung auf die gleiche Art und Weise wie eine UTM-Anwendung, siehe Seite 21f. Sie müssen nur die folgenden zusätzlichen Optionen einstellen:

- Bei den Optionen des Developer Studios müssen Sie das Verzeichnis angeben, in denen die Datenbank-Bibliotheken stehen, siehe Seite 23.
- Bei den Binderoptionen müssen Sie die Namen der benötigten Datenbank-Bibliotheken angeben, siehe Seite 27.

Damit werden beim Aufruf des Binders die richtigen Datenbank-Bibliotheken dazugebunden, siehe Seite 29.

Sie können sich mit Hilfe des Quick Start Kit eine UTM-Datenbank-Anwendung erzeugen, siehe Seite 211. Das Quick Start Kit wird mit *open*UTM ausgeliefert.

6.4 UTM-Datenbank-Anwendung starten und beenden

Eine UTM-Datenbank-Anwendung wird wie eine UTM-Anwendung gestartet und beendet, d.h. durch Starten und Beenden des UTM-Anwendungsprogramms.

6.4.1 Startparameter für eine UTM-Datenbank-Anwendung

Für den Start einer UTM-DB-Anwendung müssen neben den Startparametern für *open*UTM auch die Startparameter für die Datenbank angegeben werden. Dabei gilt folgendes Schema:

`.UTM` ...

Startparameter für *open*UTM, siehe Abschnitt „Startparameterdatei der Anwendung“ auf Seite 58.

...

`.RMXA` ...

Startparameter für das Datenbanksystem. Diese sind im Handbuch für das DB-System beschrieben.

`END`

Startparameter für das Datenbanksystem haben das Präfix `.RMXA`. *open*UTM leitet dann beim Start der Anwendung diese Startparameter an den Resource Manager weiter. Der Resource Manager wird während der Startphase des UTM-Workprozesses von *open*UTM geöffnet.

Openstring und Closestring

In der Startparameterdatei definieren Sie die Datenbank (Instanz des Resource Managers) durch einen Openstring und, falls der Resource Manager diesen benötigt, einen Closestring. Die Datenbanksysteme INFORMIX V7.2 sowie ORACLE ab V7.3 benötigen **keinen** Closestring.

Die Angaben für Openstring und Closestring sind vom jeweiligen Resource Manager zur Verfügung zu stellen. Damit hängt auch die Syntax dieser Angaben vom jeweiligen Resource Manager ab und ist dem Handbuch zu dem verwendeten Resource Manager zu entnehmen. *open*UTM übergibt die Strings aus der Startparameterdatei an den Resource Manager, ohne sie zu überprüfen. Jeder String muss durch doppelte Hochkommata begrenzt werden und darf maximal 255 Zeichen lang sein.

Bei der Kopplung über die XA-CAE-Schnittstelle werden Openstring und Closestring in **einer** Zeile der Startparameterdatei angegeben, durch ein Leerzeichen voneinander getrennt (Closestring nur, wenn vom Resource Manager benötigt):

```
.RMXA RM="name", OS="openstring" [CS="closestring"]
```

Die Zeile darf insgesamt maximal 560 Zeichen lang sein.

Mehrere Instanzen

Bei Kopplung über die XA-CAE-Schnittstelle kann die UTM-Anwendung mehrere Instanzen (Datenbanken) des Resource Managers betreiben, sofern der Resource Manager den Multi-Instanzen-Betrieb unterstützt (z.B. ORACLE V7.3). Dazu müssen Sie für jede Instanz einen eigenen Openstring angeben. Jeder Openstring ist in einer eigenen Zeile in der Startparameterdatei anzugeben. Der Name des Resource Managers *name* muss in den einzelnen Startparameter-Anweisungen übereinstimmen. Für die Openstrings (Datenbanken) sind verschiedene Namen anzugeben (angegeben im Parameter DB= innerhalb der Strings).

```
.RMXA RM="name", OS="openstring1"
.RMXA RM="name", OS="openstring2"
```

Benötigt der Resource Manager einen Closestring, dann ist für jede Instanz zusätzlich ein Closestring anzugeben.

ACHTUNG!

Neben einem gekoppeltem Datenbankanschluss darf es keinen ungekoppelten Datenbankanschluss geben.

Im Folgenden finden Sie Beispiele für Startanweisungen an die einzelnen Datenbanksysteme, mit denen *open*UTM gekoppelt werden kann.

Beispiel für INFORMIX-Startparameter in UNIX

INFORMIX benötigt nur einen Openstring, keinen Closestring. Ein Multi-Instanzen-Betrieb ist nicht möglich, d.h in der Startparameterdatei einer UTM-Anwendung darf nur ein Openstring (d.h. eine Startparameteranweisung) für den Resource Manager angegeben werden.

In der Startparameterdatei können Sie für eine INFORMIX-Datenbank z.B. die folgenden Startparameter angeben:

```
.RMXA RM="INFORMIX-ONLINE", OS="dbname"
```

Geben Sie zusätzlich `.RMXA DEBUG=YES` in der Startparameterdatei an, dann werden DEBUG-Informationen für den Anschluss an die Datenbank auf *stdout* und *stderr* ausgegeben.

Bedeutung der Parameter des Openstrings:

INFORMIX-ONLINE
: Von INFORMIX vorgegebener Name des Resource Managers, wie in der *xa_switch*-Struktur enthalten.

dbname
: Name der Datenbank, mit der die Anwendung gekoppelt werden soll.

  Pflichtparameter

Sehen Sie hierzu auch die Dokumentation zu INFORMIX.

Beispiel für ORACLE-Startparameter

ORACLE benötigt nur einen Openstring, keinen Closestring.

Ein Multi-Instanzen-Betrieb ist möglich, d.h in der Startparameterdatei einer UTM-Anwendung dürfen mehrere Openstrings für den Resource Manager angegeben werden.

In der Startparameterdatei können Sie für eine ORACLE-Datenbank z.B. die folgenden Startparameter angeben:

```
.RMXA Oracle_XA OS="Oracle_XA+Acc=P//+SesTm=60"
```

Diese Anweisung müssen Sie in eine Zeile ohne Zeilenvorschub schreiben.

Im Openstring sind nur die Pflichtparameter aufgeführt. Zusätzlich können Sie weitere optionale Parameter angeben. Diese sind weiter unten aufgelistet.
Die Parameter im Openstring werden durch das Zeichen "+" getrennt.

Bedeutung der Pflichtparameter:

Oracle_XA
: Von ORACLE vorgegebener Name des Resource Managers, wie in der *xa_switch*-Struktur enthalten.

Acc=P//
: Informationen für die Zugriffskontrolle der Datenbank (User Access Information). Wird wie im Beispiel `Acc=P//` angegeben, so werden Benutzerkennung und -Kennwort zur Zugriffskontrolle übergeben.
  Eine ORACLE-Datenbank kann auch datenbankspezifische Informationen (Benutzer *user* und Kennwort *pwd*) anfordern, die mit `Acc=P/`*user*`/`*pwd* übergeben werden müssen. Informationen dazu entnehmen Sie bitte dem Handbuch zu ORACLE.

SesTm=
: Maximale Zeit in Sekunden, die für eine Transaktion zur Verfügung steht (im Beispiel 60 s). Mögliche Angaben für *SesTm* entnehmen Sie bitte dem Handbuch zu ORACLE.

  i Die Angabe von `SesTM=0` bedeutet, dass die Transaktionsdauer nicht beschränkt wird. Es wird deshalb empfohlen, für *SesTm* einen Wert > 0 anzugeben.

Die im Folgenden aufgelisteten Parameter sind optional. Ihre Bedeutung und die möglichen Angaben entnehmen Sie bitte dem Handbuch zu ORACLE.

DB= Name der ORACLE-Datenbank

Den Parameter müssen Sie angeben, wenn die UTM-Anwendung mit mehreren ORACLE-Datenbanken gekoppelt werden soll (Multi-Instanzen-Betrieb).

GPwd=P/ Gruppenkennwort, wird in der Form `GPwd=P/`*kennwort* angegeben

LogDir= Pfadname des Logging-Dateiverzeichnisses

MaxCur= Maximale Anzahl offener Cursor

SqlNet= Network Connection String

Bei abnormalem Anwendungsende sind keine Maßnahmen notwendig, denn vor dem erneuten Start der UTM-Anwendung führt *open*UTM automatisch eine gemeinsame Recovery Phase durch.

Beachten Sie, dass die RMXA-Anweisung bei der Generierung mit KDCDEF den Operand DLLIMPORT=YES benötigt. Informationen über die einzubindenden Objekte bzw. Bibliotheken und Startparameter (Openstring) entnehmen Sie bitte der Dokumentation zu ORACLE.

Startparameter für MS-SQL-Server in Windows

Für die XA-Kopplung zwischen einer UTM-Anwendung und dem MS-SQL-Server müssen Sie folgende Zeile in die Startparameterdatei schreiben:

`.RMXA MS_SQL_Server OS="Tm=openUTM,RmRecoveryGuid=`*GUID-String*`"`

Den Wert, den Sie für *GUID-String* angeben müssen, erhalten Sie durch Aufruf des Microsoft-Programms `GUIDGEN.EXE`.

6.4.2 UTM-Datenbank-Anwendung normal beenden

Eine UTM-Datenbank-Anwendung beenden Sie mit Hilfe der UTM-Administration, siehe Abschnitt „UTM-Anwendung normal beenden“ auf Seite 69. *open*UTM schließt den Resource Manager, während sich der UTM-Workprozess der Anwendung beendet.

Wenn eine Anwendung in Windows als Dienst gestartet wurde, dann muss sie auch als Dienst beendet werden, siehe Abschnitt „Dienst beenden in Windows“ auf Seite 70.

6.4.3 UTM-Datenbank-Anwendung abnormal beenden

Eine UTM-Datenbank-Anwendung kann per Administration oder aufgrund von Fehlern abnormal beendet werden, siehe Abschnitt „UTM-Anwendung abnormal beenden“ auf Seite 71. Nach einem abnormalen Anwendungsende kann es vorkommen, dass die Datenbank oder die KDCFILE in einem nicht-konsistentem Zustand sind.

In diesem Fall wird die Konsistenz der Daten beim anschließendem Warmstart der UTM-Datenbank-Anwendung überprüft und ggf. wieder hergestellt. Dabei führt *open*UTM eine gemeinsame Recovery Phase mit den beteiligten Datenbanksystemen durch.

6.5 Betrieb einer UTM-Datenbank-Anwendung

Der Betrieb einer UTM-Datenbank-Anwendung gehorcht den selben Prinzipien wie der Betrieb einer UTM-Anwendung. Die Besonderheiten, die dabei zu beachten sind, sind in den nachfolgenden Abschnitten beschrieben.

6.5.1 Anmelden und Abmelden eine Benutzers

Ein Benutzer, der mit einer UTM-Datenbank-Anwendung arbeiten möchte, meldet sich indem er sich über die client-spezifischen Anmeldeverfahren bei *open*UTM an. Entsprechendes gilt für das Abmelden.

Beim Anmelden stehen alle Möglichkeiten offen, die UTM für das Anmelden bietet. Insbesondere kann der Benutzer die SIGNON-Services von UTM nutzen. Dabei ist Folgendes zu beachten:

– Meldet sich der Benutzer als Terminal an, dann sind Datenbank-Aufrufe im ersten Teil des SIGNON-Services aus Sicherheitsgründen nicht erlaubt, es sei denn, man lässt dies per Generierung explizit zu mit der KDCDEF-Anweisung SIGNON, ...RESTRICTED=NO.

– Im zweiten Teil des SIGNON-Services kann das Berechtigungsprofil für den Benutzer aus der Datenbank gelesen werden. Damit lässt sich ein DB/DC-übergreifendes Berechtigungskonzept realisieren.

Näheres zum An- und Abmelden finden Sie im Kapitel „Arbeiten mit einer UTM-Anwendung“ auf Seite 83.

6.5.2 Diagnose

UTM bietet zur Fehlerdiagnose bei einer UTM-Datenbank-Anwendung die gleichen Informationsquellen wie bei einer reinen UTM-Anwendung, das sind UTM-Meldungen, Fehlercodes und Dumps. Einige dieser Quellen enthalten auch datenbankspezifische Daten. Diese sollten als erstes herangezogen werden, wenn der Fehler mit der Datenbank-Kopplung zusammenhängen könnte. Dabei handelt es sich um folgende UTM-Diagnose-Informationen:

– Die datenbankspezifischen UTM-Meldungen K068 und K071

– Die Start-Fehlercodes der Meldung K049

– Den inkompatiblen Returncode KCRCDC

– Die DB-DIAGAREA des UTM-Dumps, falls ein UTM-Dump erzeugt wurde

Einzelheiten dazu finden Sie im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

7 Arbeiten mit einer UTM-Anwendung

Dieses Kapitel beschreibt, wie ein Benutzer über die unterschiedlichen Clients mit einer UTM-Anwendung kommunizieren kann. Für alle Clients verläuft die Kommunikation immer nach folgendem Prinzip:

1. Anmelden an die UTM-Anwendung
   Ein Benutzer kann sich nur über Clients anmelden, für die LTERM-Partner, LTERM-Pools oder als OSI-LPAP-Partner in der UTM-Anwendung generiert sind, siehe *open*UTM-Handbuch „Anwendungen generieren“. Eine Anmeldung z.B. über Remote-Login-Mechanismen ist nicht möglich.
2. Services (=Vorgänge) der UTM-Anwendung aufrufen:
   Für der Zugriffsschutz bietet *open*UTM ein eigenes Berechtigungskonzept, siehe Seite 109.
3. Ggf. UTM-Benutzerkommandos eingeben.
4. Abmelden von der UTM-Anwendung

Im Detail unterscheiden sich diese Schritte je nach Art des Clients. Die folgenden Abschnitte beschreiben, welche Möglichkeiten es für die verschiedenen Clients gibt.

Für den Zugang werden UTM-Benutzerkennungen verwendet, sofern die Anwendung mit Benutzerkennungen generiert ist. Das Anmelden an eine UTM-Anwendung ohne Benutzerkennungen wird auf Seite 104 behandelt.

7.1 Anmeldeverfahren mit Benutzerkennungen

Ist eine Anwendung mit Benutzerkennungen generiert, dann führt *open*UTM für den Benutzer abhängig von der Art des Client ein Standard-Anmeldeverfahren durch. Daneben besteht die Möglichkeit, anstelle des Standard-Verfahrens selbst erstellte Anmeldeverfahren zu verwenden, siehe Abschnitt „Anmeldeverfahren mit Anmeldevorgängen“ auf Seite 98.

Ein Benutzer kann sich über folgende Client-Zugänge anmelden:

– Terminals (siehe unten)

– UPIC-Clients und TS-Anwendungen (Seite 93)

– OSI-TP-Clients (Seite 95)

– über das Web mit Hilfe von Web*Transactions* (Seite 96)

Grundsätzlich können sich auch mehrere Benutzer unter einer Benutzerkennung anmelden, siehe Abschnitt „Mehrfach-Anmeldungen unter einer Benutzerkennung“ auf Seite 97.

7.1.1 Standard-Anmeldeverfahren für Terminals

Der Benutzer führt folgende Schritte durch, um über ein Terminal mit einer UTM-Anwendung zu arbeiten:

1. Anmelden bei UNIX oder Windows
2. Starten des Dialogterminalprozesses, siehe Seite 85. Dabei meldet sich der Benutzer gleichzeitig an die UTM-Anwendung an.

Erst dann kann der Benutzer Vorgänge starten und im Dialog bearbeiten, siehe Abschnitt „UTM-Services aufrufen“ auf Seite 105.

Mit der Umgebungsvariablen LANG kann der Benutzer das Verhalten der Anwendung beeinflussen:

Mit LANG gibt er an, in welcher Sprache die an ihn gerichteten UTM-Meldungen ausgegeben werden. Es werden standardmäßig die Einstellungen LANG=De... für die deutschen Meldungen und LANG=En... für die englischen Meldungen unterstützt. Die Meldungen werden aus NLS-Meldungskatalogen aufgebaut.

7.1.1.1 Starten des Dialogterminalprozesses durch den Benutzer

Standardmäßig muss der Benutzer den Dialogterminalprozess selbst starten, nachdem er sich bei UNIX bzw. Windows angemeldet hat.

- Unter UNIX gibt er dazu folgendes Kommando ein:

  utm-dateiverzeichnis`/ex/utmdtp [-S[`*username*`]] [-A`*applicationname*`] [-P`*ptermname*`] [-D]`

  Es gibt auch die Möglichkeit, den Dialogterminalprozess durch UNIX starten zu lassen, siehe Seite 87.

- Unter Windows muss er an der Console mit *Start - Programme - Eingabeaufforderung* ein DOS-Fenster öffnen und folgendes Kommando eingeben:

  utm-dateiverzeichnis`\ex\utmdtp [-S[`*username*`]] [-A`*applicationname*`] [-P`*ptermname*`]`

  Das DOS-Fenster darf erst geschlossen werden, wenn der Benutzer sich von der Anwendung abgemeldet hat.

Erläuterung

Die Angaben in eckigen Klammern stellen Schalter dar, die man angeben kann, aber nicht angeben muss. Die Schalter haben folgende Bedeutung:

-S[*username*]

Mit diesem Schalter steuert der Benutzer die Berechtigungsprüfung (Zugangskontrolle) im Dialog, die *open*UTM nach dem erfolgreichen Verbindungsaufbau zur UTM-Anwendung durchführt.

Dialogterminalprozess **mit** Schalter -S:
Startet der Benutzer den Dialogterminalprozess mit Schalter -S, so muss er eine UTM-Benutzerkennung *username* zur Berechtigungsprüfung an *open*UTM übergeben. Mit -S*username* kann der Benutzer die UTM-Benutzerkennung direkt beim Start des Dialogterminalprozesses angeben. Mit -S ohne *username* erfragt *open*UTM die UTM-Benutzerkennung nach dem Verbindungsaufbau im Dialog.

Gibt der Benutzer eine UTM-Benutzerkennung an, für die ein Passwort oder eine Ausweisüberprüfung generiert ist, dann fragt *open*UTM die entsprechenden Daten im Dialog ab, siehe Beschreibung ab Seite 88.

Dialogterminalprozess **ohne** Schalter -S:
Startet der Benutzer den Dialogterminalprozess ohne Schalter -S, so übergibt der Dialogterminalprozess die UNIX- bzw. Windows-Benutzerkennung zur Berechtigungsprüfung. Das UNIX-/Windows-Passwort wird nicht an *open*UTM übergeben. In der UTM-Anwendung darf für die Benutzerkennung ein Passwort vergeben werden, zu dessen Eingabe - wie im Fall der expliziten Angabe der Benutzerkennung - der Benutzer aufgefordert wird.

Verläuft eine Prüfung mit der UNIX-/Windows-Benutzerkennung negativ, dann folgt ein expliziter Berechtigungsdialog wie bei der Verwendung des Schalters -S.

Wie der Berechtigungsdialog abläuft, ist in Abschnitt „Standard-Anmeldedialog“ auf Seite 88 beschrieben.

Schalter -S und *username* bilden einen String und dürfen durch kein Zeichen getrennt werden.

-A*applicationname*

Mit diesem Schalter gibt der Benutzer die Anwendung an, mit der er verbunden werden will. *applicationname* ist der Name der Anwendung. Wird -A*applicationname* beim Start des Dialogterminalprozesses **nicht** angegeben, dann wird der Benutzer im Zeilenmodus zur Eingabe des Anwendungsnamens aufgefordert.

Schalter -A und *applicationname* bilden einen String und dürfen durch kein Zeichen getrennt werden.

-P*ptermname*

ptermname ist der Name des Terminals, über das der Benutzer die Verbindung zu *open*UTM aufbaut. Dieser Name muss in einer PTERM-Anweisung generiert sein oder es muss ein LTERM-Pool für lokale Terminals definiert sein (TPOOL PTYPE=TTY,LTERM=*ltermprefix*,NUMBER=*number*). Wenn ein solcher LTERM-Pool generiert ist, kann der Schalter -P immer weggelassen werden. Andernfalls gilt:

- Falls in UNIX kein solcher LTERM-Pool generiert ist, dann kann der Schalter -P bis auf den unten beschriebenen Ausnahmefall weggelassen werden, denn standardmäßig verwendet *open*UTM unter UNIX als *ptermname* den letzten Teil der Ausgabe des tty-Kommandos. Das ist der Term hinter dem letzten Schrägstrich, er entspricht der Ausgabe von `basename `tty``.

  Der Schalter -P ist nur dann notwendig, wenn die Standardzuordnung von *ptermname* durch *open*UTM nicht eindeutig ist, z.B. wenn mehrere Pseudoterminals existieren, die sich in dem letzten Term des tty (hinter dem letzten Schrägstrich) nicht unterscheiden.

  Beispiel
  Es existieren gleichzeitig die ttys `/dev/pts/12` und `/dev/inet/12`. Diese ttys sind generiert mit PTERM pts/12 und PTERM inet/12. In diesem Fall muss der Benutzer den Dialogterminalprozess mit Schalter -P starten, z.B. wie folgt:
  utm-dateiverzeichnis`/ex/utmdtp` ... `-Ppts/12`.

- Falls in Windows kein solcher LTERM-Pool generiert ist, dann muss der Benutzer immer -P*ptermname* angeben, denn *open*UTM verwendet unter Windows für *ptermname* standardmäßig den (nicht generierten) Wert `tty`*nnnnn*, *nnnnn*=`PID`.

Beispiel
Wenn das lokale Terminal mit PTERM console,PTYPE=TTY,... generiert ist, dann muss er den Dialogterminalprozess starten mit
utm-dateiverzeichnis\ex\utmdtp ... -Pconsole.

Schalter -P und *ptermname* bilden einen String und dürfen durch kein Zeichen getrennt werden.

-D

Der Benutzer bestimmt unter UNIX mit diesem Schalter die Reaktion des Dialogterminalprozesses auf die Taste DEL.

Schalter -D angegeben: Die Taste DEL wird vom Dialogterminalprozess ignoriert.

Schalter -D **nicht** angegeben: Drücken der Taste DEL führt zum Abmelden von der Anwendung und zur Beendigung des Dialogterminalprozesses.

Beispiel
Der Benutzer möchte sich unter der UTM-Benutzerkennung user1 an die Anwendung sample anmelden; die Taste DEL soll ignoriert werden. Dazu muss er folgendes Kommando eingeben:

utm-dateiverzeichnis/ex/utmdtp -Suser1 -Asample -D

7.1.1.2 Starten des Dialogterminalprozesses durch UNIX

UNIX kann den Dialogterminalprozess auch von sich aus starten, nachdem der Benutzer sich erfolgreich bei UNIX angemeldet hat. Dazu kann er das Kommando zum Starten des Dialogterminalprozesses z.B. wie folgt in seine eigene .profile eintragen:

– .../utmdtp *schalter* bewirkt, dass nach Beendigung von utmdtp die Shell noch aktiv bleibt.

– exec .../utmdtp *schalter* bewirkt, dass nach Beendigung von utmdtp die Shell ebenfalls beendet wird.

Eine zweite Möglichkeit besteht darin, dass der Systemverwalter in der Datei /etc/passwd für den Benutzer als Programmnamen den Dialogterminalprozess einträgt oder (bei Verwendung von Schaltern) eine Shell-Prozedur startet, in welcher der Aufruf des Dialogterminalprozesses steht.

Der Dialogterminalprozess ist sinnvollerweise erst dann zu starten, wenn die Workprozesse den erfolgreichen Kalt- oder Warmstart der Anwendung gemeldet haben. Sonst beendet sich der Dialogterminalprozess wieder mit der Meldung:

U111 UTM-Anwendung *applicationname* ist nicht gestartet.

Tritt ein Fehler auf beim Ablauf des Dialogterminalprozesses, beendet sich dieser mit der folgenden Meldung. Die Fehlernummern *nnnn* sind im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“ beschrieben:

```
U120 utmdtp-Prozeß beendet sich wegen Fehlernummer nnnn.
```

7.1.1.3 Standard-Anmeldedialog

Der Standard-Anmeldedialog wird immer dann durchgeführt, wenn die beiden folgenden Bedingungen erfüllt sind:

1. Für das Terminal ist kein automatisches KDCSIGN (= automatische Berechtigungsprüfung) generiert (siehe Seite 93).
2. Für den Anwendungsnamen, unter dem sich der Benutzer angemeldet hat, ist kein Anmeldevorgang generiert (siehe Seite 98).

Beim Standard-Anmeldedialog führt *open*UTM eine Berechtigungsprüfung durch (Zugangskontrolle). Der Anmeldedialog ist nicht modifizierbar.

Die Berechtigungsprüfung kann unterschiedlich streng festgelegt werden. Eine Übersicht aller Möglichkeiten enthält die Abbildung in Abschnitt „Szenarien für die UTM-Berechtigungsprüfung“ auf Seite 90.

Die Berechtigungsprüfung führt *open*UTM im Dialog mit dem Benutzer durch, wenn beim Start des entsprechenden Dialogterminalprozesses der Schalter `-S` angegeben wird. In diesem Fall fragt *open*UTM die Benutzerkennung ab und fordert - falls generiert- die Angabe eines Passworts an.

Wird der Schalter `-S` nicht angegeben, führt *open*UTM die Berechtigungsprüfung mit der UNIX- bzw. Windows-Benutzerkennung durch, unter der sich der Benutzer angemeldet hat. In diesem Fall kann in der UTM-Anwendung für die Benutzerkennung ebenfalls ein Passwort generiert werden, zu dessen Eingabe der Benutzer bei der Berechtigungsprüfung aufgefordert wird.

Zu beachten ist jedoch, dass ein Benutzer nicht unter einer Benutzerkennung gleichzeitig mit mehreren Dialogterminalprozessen arbeiten kann.

Die Benutzerkennung wird im Zeilenmodus angefordert. Zur Eingabe eines Passworts siehe nächster Abschnitt.

Eingabe des Passworts

Wenn für die Benutzerkennung ein **Passwort** generiert ist (KDCDEF-Anweisung USER...,PASS=*password*[, DARK]), dann wird das Passwort immer in ein Feld ohne Dunkelsteuerung eingegeben.

Bei jeder UTM-Anwendung hat der Benutzer die Möglichkeit, ein neues Passwort einzugeben, welches das bisherige ersetzen soll, vorausgesetzt, die minimale Gültigkeitsdauer erlaubt die Passwortänderung zu diesem Zeitpunkt. Das neue Passwort muss dann einmal in ein Feld ohne Dunkelsteuerung eingegeben werden. *open*UTM prüft das angegebene alte Passwort und ggf. das neue Passwort. Ist das alte Passwort nicht richtig oder stimmen die Angaben zum neuen Passwort nicht überein, dann wird der Benutzer mit einer UTM-Meldung informiert und zur Wiederholung der Eingabe aufgefordert.

Gültigkeitsdauer des Passwortes

Bei der Generierung der Benutzerkennung kann eine maximale und eine minimale Gültigkeitsdauer für das Passwort vereinbart werden:

USER ...,PROTECT-PW=(...,*maxtime*,*mintime*)

Die minimale Gültigkeitsdauer bedeutet, dass der Benutzer nach einer Passwort-Änderung die nächste Änderung erst nach Ablauf der festgelegten Zeit vornehmen kann. Die maximale Gültigkeitsdauer bedeutet, dass der Benutzer das Passwort jeweils innerhalb der angegebenen Zeitspanne ändern muss.

Wird das Passwort innerhalb der nächsten 14 Tage nach der Anmeldung ungültig, warnt *open*UTM den Benutzer mit einer K-Meldung, sofern die minimale Gültigkeitsdauer des Passworts eine Änderung zu diesem Zeitpunkt erlaubt. Ein dunkelgesteuertes Passwort kann der Benutzer wie unter „„Eingabe des Passworts“ auf Seite 88“ beschrieben ändern. Um zu verhindern, dass Benutzer, die längere Zeit nicht mit der Anwendung arbeiten, die Passwort-Änderung versäumen und sich dann an den Administrator wenden müssen, kann die UTM-Anwendung so konfiguriert werden, dass auch nach Ablauf des Passworts eine Anmeldung erlaubt wird, siehe nächste Seite, Abschnitt „Grace-Sign-On“.

*open*UTM überprüft bei der Passwortänderung, ob:

- das neue Passwort sich von dem alten Passwort unterscheidet.
  Wenn eine Passwort-Historie generiert ist (SIGNON ...,PW-HISTORY=*n*), dann wird auch gegen die letzten *n* Passworte geprüft.
- das neue Passwort der Komplexitätsstufe entspricht, die für die Benutzerkennung generiert wurde (USER ...,PROTECT-PW=),
- die Länge des Passwortes größer oder gleich der generierten Mindestlänge ist (USER ...,PROTECT-PW=).

Erfüllt das Passwort alle diese Punkte, nimmt *open*UTM die Passwortänderung vor. Die Gültigkeitsdauer des neuen Passwortes entspricht wieder dem generierten Wert.
Erfüllt das neue Passwort einen dieser Punkte nicht, so wird der Benutzer mit der folgenden Meldung aufgefordert, die KDCSIGN-Eingabe mit dem alten Passwort zu wiederholen:

```
K097 Angaben zum neuen Passwort sind nicht verwendbar - Bitte Anmelden
```

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist kein Grace-Sign-On generiert, so wird die Anmeldung mit der folgenden Meldung zurückgewiesen:

```
K120 Die Gueltigkeit des Passworts ist abgelaufen
```

Diese Benutzerkennung kann sich erst wieder bei der UTM-Anwendung anmelden, wenn der UTM-Administrator der Kennung mit dem Kommando KDCUSER ein neues Passwort zugewiesen hat.

Grace-Sign-On

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist die Anwendung ist mit Grace-Sign-On generiert (SIGNON ...,GRACE=YES), so wird der Benutzer mit einer K-Meldung darauf hingewiesen, dass sein Passwort nicht mehr gültig ist. Gleichzeitig wird er aufgefordert, sein bisheriges Passwort und ein neues Passwort einzugeben.

Szenarien für die UTM-Berechtigungsprüfung

Das folgende Diagramm zeigt, welche Varianten der UTM-Berechtigungsprüfung möglich sind - abhängig von der KDCDEF-Generierung. Bei fehlerhaften Eingaben gibt *open*UTM eine spezifische Meldung aus und fordert den Benutzer zu einer neuen Eingabe auf. Werden von einem bestimmten Terminal aus nacheinander mehrere erfolglose Anmeldeversuche unternommen, dann erzeugt *open*UTM die Meldung K094 mit dem Standardziel SYSLOG (System-Protokolldatei). Die maximal zulässige Anzahl erfolgloser Anmeldeversuche bis zum Auslösen der Meldung K094 kann mit SIGNON ... SILENT-ALARM= bei der KDCDEF-Generierung festgelegt werden. Ein MSGTAC-Teilprogramm kann auf diese Meldung reagieren.

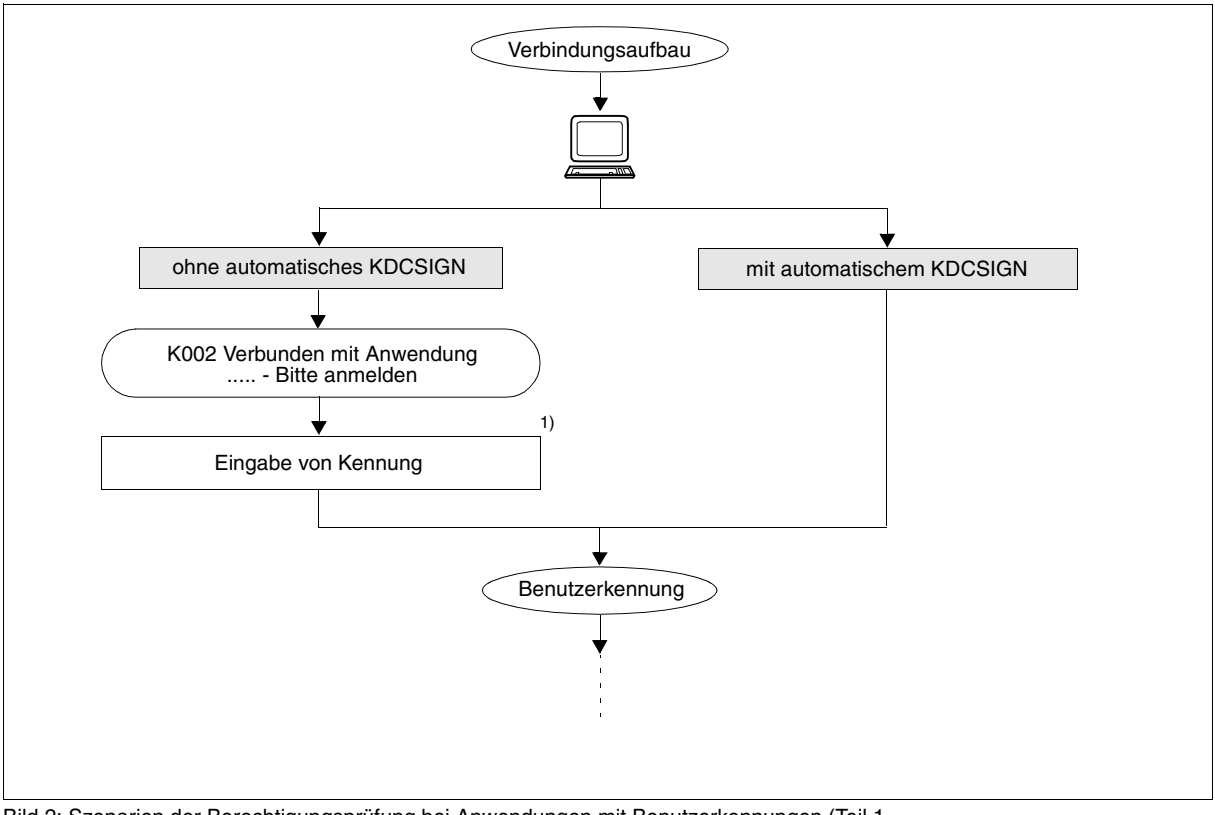
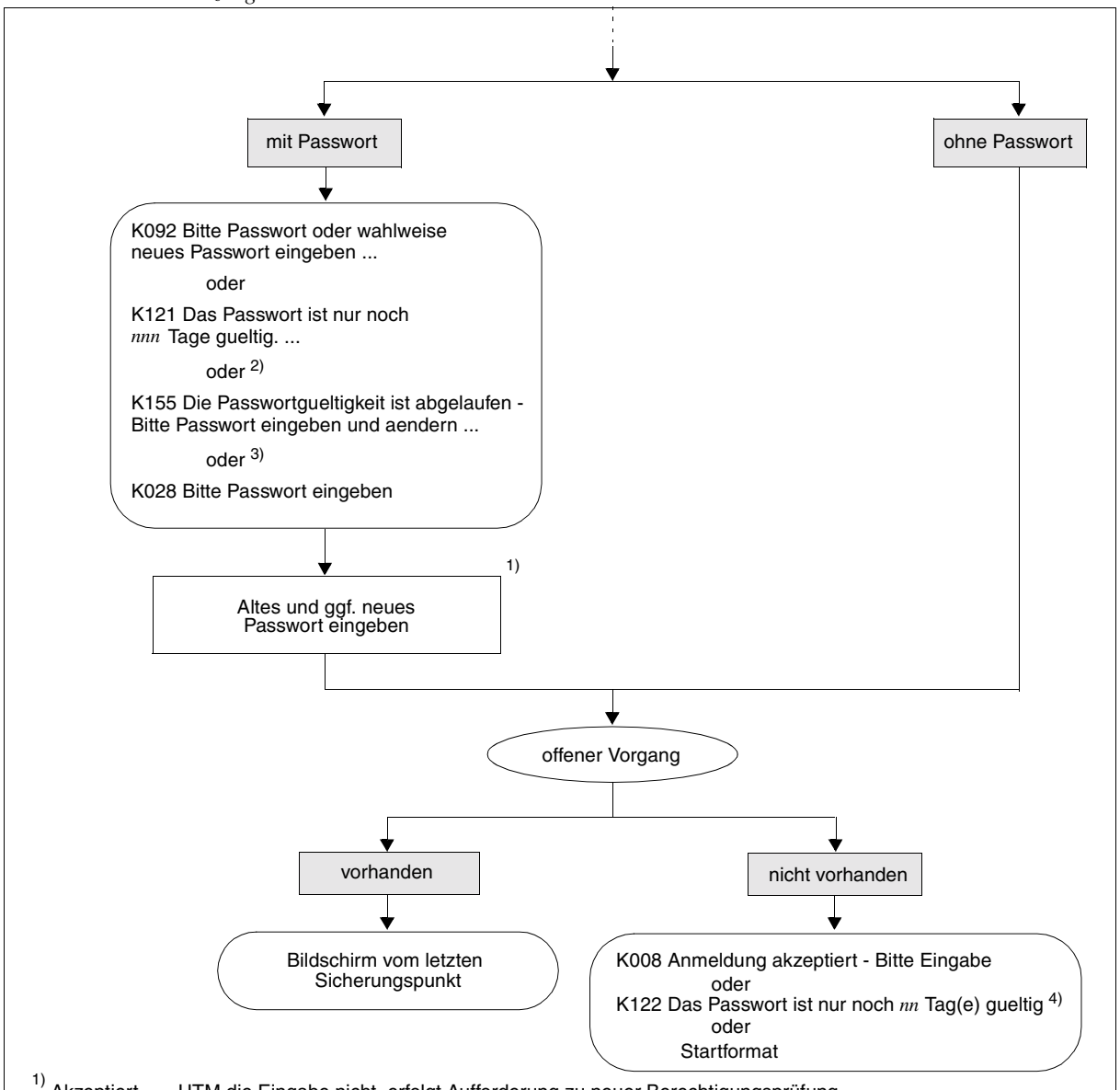


Bild 2: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 1

Fortsetzung:


mit Passwort
ohne Passwort
K092 Bitte Passwort oder wahlweise neues Passwort eingeben ...
oder
K121 Das Passwort ist nur noch nnn Tage gueltig. ...
oder 2)
K155 Die Passwortgueltigkeit ist abgelaufen - Bitte Passwort eingeben und aendern ...
oder 3)
K028 Bitte Passwort eingeben
1)
Altes und ggf. neues Passwort eingeben
offener Vorgang
vorhanden
nicht vorhanden
Bildschirm vom letzten Sicherungspunkt
K008 Anmeldung akzeptiert - Bitte Eingabe
oder
K122 Das Passwort ist nur noch nn Tag(e) gueltig 4)
oder
Startformat

1) Akzeptiert *open*UTM die Eingabe nicht, erfolgt Aufforderung zu neuer Berechtigungsprüfung.
2) Falls das Passwort abgelaufen und eine Änderung erlaubt ist.
3) Falls Änderung des Passworts nicht erlaubt ist.
4) Bei Benutzern mit Passwort, wenn das Passwort weniger als 14 Tage gültig ist und wenn Änderung erlaubt ist und wenn kein Startformat für den Benutzer generiert ist

Bild 2: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 2)

7.1.1.4 Automatisches KDCSIGN

Wenn für ein Terminal die KDCDEF-Anweisung LTERM...,USER=*username* angegeben wurde, dann verhält sich *open*UTM nach dem Verbindungsaufbau so, als ob der Benutzer seine Berechtigung schon erfolgreich nachgewiesen hätte. Wenn für diese Benutzerkennung die Eingabe eines Passworts vorgeschrieben ist, dann fordert *open*UTM diese Eingabe(n) vom Benutzer an.

Nach Eingabe von KDCOFF BUT kann man an diesem Terminal auch unter einer anderen Kennung arbeiten (siehe Abschnitt „Abmelden von der UTM-Anwendung“ auf Seite 111).

7.1.2 Anmeldeverfahren für UPIC-Clients und TS-Anwendungen

UPIC-Clients und TS-Anwendungen sind Clients, die mit PTYPE=UPIC-L, UPIC-R, APPLI oder SOCKET generiert wurden.

Bei UPIC-Clients wird die Verbindung durch den Client und bei TS-Anwendungen durch den Client oder durch UTM aufgebaut, wobei der Verbindungsaufbau durch UTM nur möglich ist, wenn die TS-Anwendung explizit mit einer PTERM-Anweisung generiert ist.

Falls der Client die Verbindung aufbaut, muss dieser den Namen der UTM-Anwendung sowie Rechnernamen und/oder Rechneradresse kennen. Diese Daten werden auf dem Client-Rechner konfiguriert, z.B. im Transport Name Server (TNS) von CMX/PCMX.

Nach dem erfolgreichen Verbindungsaufbau wird ein UPIC-Client oder eine TS-Anwendung in zwei Schritten angemeldet:

1. Implizites Anmelden über eine **Verbindungs-Benutzerkennung**

   Eine Verbindungs-Benutzerkennung ist einem LTERM-Partner fest zugeordnet und wird für jede TS-Anwendung und jeden UPIC-Client bei der Generierung explizit oder implizit erzeugt:

   – Explizit durch die Angabe bei USER= in der LTERM-Anweisung.
   Für eine so definierte Verbindungs-Benutzerkennung können mit der KDCDEF-Anweisung USER weitere Eigenschaften festgelegt werden.

   – Implizit durch *open*UTM, wenn in der LTERM-Anweisung kein USER angegeben wurde oder wenn es sich um einen LTERM-Pool handelt (TPOOL-Anweisung). Als Verbindungs-Benutzerkennung wird dann der LTERM-Name genommen; bei einem LTERM-Pool setzt sich der LTERM-Name aus dem generierten Präfix und eine laufenden Nummer zusammen, z.B. UPIC0025. Für LTERM-Pools können der Verbindungs-Benutzerkennung mit TPOOL ...USER-KSET= spezielle Keycodes zugeordnet werden. Damit lassen sich die Zugriffsmöglichkeiten der Verbindungs-Benutzerkennung einschränken.

Folgt keine Anmeldung unter einer echten Benutzerkennung, wird die vorläufige Anmeldung der Verbindungs-Benutzerkennung zu einer endgültigen. Dies wird mit einer Meldung protokolliert. Bei UPIC-Clients wird diese Meldung auch ausgegeben, wenn sich dieser anschließend unter einer echten Benutzerkennung anmeldet.

2. Explizites Anmelden über eine **echte Benutzerkennung** (optional)

   Das Verhalten von UPIC-Clients und TS-Anwendungen ist dabei unterschiedlich:

   – Bei UPIC-Clients müssen die Benutzerkennung und die Berechtigungsdaten in den jeweiligen UPIC-Schnittstellenaufrufen gesetzt werden, UPIC übergibt diese Werte anschießend an *open*UTM. *open*UTM führt dann die Anmeldung für die übergebene Benutzerkennung durch. Diese ersetzt die Verbindungs-Benutzerkennung für die Dauer der Conversation. Am Ende der Conversation wird der Benutzer wieder abgemeldet.

     Übergibt der UPIC-Client in den UPIC-Schnittstellenaufrufen keine Berechtigungsdaten, dann ist das Anmelden über eine echte Benutzerkennung nur mit einem entsprechenden Anmeldevorgang möglich, siehe Seite 98.

   – Über eine Transportsystemverbindung kann sich ein Benutzer unter einer echten Benutzerkennung nur dann anmelden, wenn für die Anwendung ein entsprechender Anmeldevorgang generiert ist, siehe Seite 98. Eine Anmeldung mit einer echten Benutzerkennung über den Standard-Anmeldedialog ist nicht möglich.

     Wurde eine TS-Anwendung über eine echte Benutzerkennung angemeldet, dann ersetzt diese Benutzerkennung die Verbindungs-Benutzerkennung für die gesamte Dauer der Verbindung.

   Sowohl für UPIC-Clients als auch für TS-Anwendungen bleibt die Verbindungs-Benutzerkennung mindestens so lange angemeldet wie die echte Benutzerkennung. Tritt ein Verbindungsverlust auf, kann das dazu führen, dass ein erneuter Verbindungsaufbau abgelehnt wird, wenn unter der echten Benutzerkennung noch ein Programm abläuft und deshalb auch die Verbindungs-Benutzerkennung als angemeldet gilt. In diesem Fall muss der Benutzer mit einer Neuanmeldung warten, bis das Programm beendet ist.

7.1.3 Anmeldeverfahren für OSI-TP-Clients

Damit sich ein OSI-TP-Client an die UTM-Anwendung anmelden kann, muss der Client den Namen der UTM-Anwendung, sowie Rechnernamen und/oder Rechneradresse kennen. Diese Daten werden auf dem Client-Rechner konfiguriert.

Bei OSI-TP-Clients kann die Initiative zum Verbindungsaufbau sowohl vom Client als auch von *open*UTM kommen. Dabei können über eine logische Verbindung mehrere parallele Verbindungen aufgebaut werden, die Associations genannt werden. Jeder Association ist ein Association-Name zugeordnet.

Nach erfolgreichem Verbindungsaufbau wird der Client zunächst unter seinem Association-Namen angemeldet. Dieser wird zusammengesetzt aus dem in OSI-LPAP ...,ASSOCIATION-NAMES= angegebenen Namen sowie der laufenden Nummer, z.B. ASSOC03.

Wenn der Client in den jeweiligen Protokollfeldern eine **echte** Benutzerkennung und Berechtigungsdaten übergibt, dann führt *open*UTM anschließend die Anmeldung für die übergebene Benutzerkennung durch. Voraussetzung dafür ist, dass in der OSI-LPAP-Anweisung der entsprechende APPLICATION-CONTEXT ist generiert. Wird keine echte Benutzerkennung übergeben, dann bleibt der Client unter seinem Association-Namen angemeldet.

Am Ende der Conversation wird der Benutzer wieder abgemeldet.

7.1.4 Anmeldeverfahren im World Wide Web über Web*Transactions*

Eine UTM-Anwendung kann über Web*Transactions* an das World Wide Web angeschlossen werden. Damit kann ein Benutzer über einen Browser auf die Services einer UTM-Anwendung zugreifen.

Die Anmeldung über Web*Transactions* ist einfach; sie kann durch die Web*Transactions*-Anwendung gestaltet werden:

1. Der Benutzer gibt in seinem Browser die URL der Web*Transactions*-Anwendung an. Danach wird die Verbindung zur UTM-Anwendung aufgebaut. Eventuell führt die Web*Transactions*-Anwendung zuvor einen eigenen Zwischendialog, z.B. für einen Berechtigungsnachweis für den Zugang zur Web*Transactions*-Anwendung.
2. Der Benutzer muss eventuell wie beim Terminal die UTM-Benutzerkennung und ggf. das Passwort angeben. Ob der Benutzer einen derartigen Berechtigungsdialog führen muss und wie dieser aussieht, hängt jedoch von der Gestaltung der Web*Transactions*-Anwendung ab. Es ist z.B. möglich, die UTM-Benutzerkennung/-Passwort in der Web*Transactions*-Anwendung zu „verstecken“, so dass der Berechtigungsdialog intern abläuft und der Benutzer nach der Eingabe der URL sofort angemeldet ist.

Anschließend kann der Benutzer die Services der Anwendung aufrufen, siehe Seite 105f.

Web*Transactions* nutzt für den Anschluss an die UTM-Anwendung die UPIC-Schnittstelle von UTM. Nähere Informationen finden Sie im Web*Transactions*-Handbuch „Anschluss an *open*UTM-Anwendungen über UPIC“.

7.1.5 Mehrfach-Anmeldungen unter einer Benutzerkennung

Ist die Benutzerkennung bei der KDCDEF-Generierung mit RESTART=NO und dem Standardwert MULTI-SIGNON=YES erzeugt worden, dann kann ein Benutzer über verschiedene Verbindungen mehrfach bei *open*UTM angemeldet sein, allerdings nur einmal über eine Verbindung zum Terminal. Mehrfach-Anmeldungen sind nur für echte Benutzerkennungen möglich, **nicht** jedoch für Verbindungs-Benutzerkennungen. Näheres zu Verbindungs-Benutzerkennungen finden Sie auf Seite 93.

Der aktuelle Vorgangskontext belegt in diesem Falle die Ressourcen der Verbindungs-Benutzerkennung (UPIC, TS-Anwendung) bzw. der Association (OSI-TP-Partner).

Meldet sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung über einen OSI TP-Partner an, für dessen Conversation die Functional Unit „Commit“ ausgewählt ist, ist unter dieser Benutzerkennung ebenfalls eine weitere Anmeldung möglich, weil *open*UTM in diesem Falle keinen Vorgangswiederanlauf durchführt und die Benutzerkennung dann so behandelt, als sei kein Wiederanlauf generiert.
Das gleiche gilt, wenn sich der Benutzer über einen OSI-TP-Partner anmeldet und einen asynchronen Auftrag ausführt.

Ansonsten kann sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung zu einem Zeitpunkt nur einmal anmelden, da die für den Vorgangswiederanlauf benötigten Ressourcen der Benutzerkennung zugeordnet werden.

Verhindern von Mehrfach-Anmeldungen für Benutzerkennungen mit RESTART=NO

Über den Parameter MULTI-SIGNON der SIGNON-Anweisung können Sie bei der Generierung festlegen, dass unabhängig von der Wiederanlaufeigenschaft ein Benutzer zu jedem Zeitpunkt nur einmal bei *open*UTM angemeldet sein darf.

Diese Festlegung gilt jedoch nicht für Anmeldungen über einen OSI-TP-Partner zur Ausführung von Asynchronaufträgen.

7.1.6 Anmeldeverfahren mit Anmeldevorgängen

Anmeldevorgänge, auch als Event-Services SIGNON bezeichnet, sind selbst programmierte Vorgänge, mit deren Hilfe eigene Anmeldeverfahren definiert werden können. Anmeldevorgänge können von Terminals, UPIC-Clients und TS-Anwendungen genutzt werden, d.h. von Clients, die über PTERM- oder TPOOL-Anweisung generiert sind.

Aufruf von Anmeldevorgängen

Ein Anmeldevorgang ist an den Anwendungsnamen gebunden. Meldet sich ein Client unter einem bestimmten Anwendungsnamen an, dann wird der zu diesem Anwendungsnamen gehörige Anmeldevorgang gestartet und ersetzt das in den vorigen Abschnitten geschilderten Standard-Anmeldeverfahren. Falls mit BCAMAPPL-Anweisungen mehrere Anwendungsnamen generiert sind, dann können mehrere unterschiedliche Anmeldevorgänge in einer Anwendung existieren. Damit lassen sich client-spezifische Anmeldevorgänge erstellen, z.B. einer für Terminals, einer für UPIC-Clients und einer für TS-Anwendungen. Weitere Details finden Sie in den Abschnitten „Anmeldevorgang für Terminals“ auf Seite 99 bis „Anmeldevorgang für UPIC-Clients“ auf Seite 100.

Ist für einen Anwendungsnamen kein Anmeldevorgang generiert, dann durchläuft der Client das Standard-Anmeldeverfahren.

Generieren von Anmeldevorgängen

Anmeldevorgänge werden wie folgt generiert, siehe auch *open*UTM-Handbuch „Anwendungen generieren“:

– Mit TAC KDCSGNTC wird der Anmeldevorgang für den Standard-Anwendungsnamen (definiert in MAX APPLINAME) generiert.

– Mit BCAMAPPL *appliname2...*,SIGNON=*signon-tac* wird der Anmeldevorgang für den Anwendungsnamen *appliname2* generiert. *signon-tac* muss in einer TAC-Anweisung definiert sein.

– Sollen Anmeldevorgänge auch von UPIC-Clients genutzt werden können, dann muss zusätzlich SIGNON ...,UPIC=YES generiert werden.

Für jeden dieser TACs ist auch eine PROGRAM-Anweisung nötig. Dort wird der Name des Teilprogramms angegeben, das im Anmeldevorgang als erstes durchlaufen wird.

Programmieren von Anmeldevorgängen

Für die Programmierung eines Anmeldevorgangs gibt es die speziellen KDCS-Aufrufe SIGN ST, SIGN ON und PEND PS. Wie ein Anmeldevorgang programmiert werden kann und welche Regeln dabei zu beachten sind, ist ausführlich im entsprechenden Abschnitt im *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

7.1.6.1 Anmeldevorgang für Terminals

Ein Anmeldevorgang für Terminals setzt sich im allgemeinen aus zwei Teilen zusammen:

Anmeldevorgang	
Teil 1	Teil 2
Der Vorgang fordert den Benutzer auf, sich zu legitimieren, liest mit MGET die Berechtigungsdaten und übergibt diese zur Prüfung an *open*UTM. Der Vorgang ist noch **keiner** Benutzerkennung zugeordnet.	*open*UTM hat die Berechtigungsdaten akzeptiert und hat den Anmeldevorgang der ermittelten Benutzerkennung zugeordnet.

Zwischen dem ersten und zweiten Teil des Anmeldevorgangs kann *open*UTM ggf. einen Zwischendialog einfügen, der über *open*UTM-Meldungen gesteuert wird. Dieser Zwischendialog dient dazu, eine Ausweiskarte einzulesen, oder ein mit USER...,PASS=(...,[DARK]) generiertes Passwort. In diesen Fällen fordert *open*UTM den Benutzer wie beim Standard-Anmeldedialog auf, diese Information einzugeben. Dieser Zwischendialog bietet dem Anwender zusätzliche Sicherheit für den Zugang zu seinen Anwendungen. Im Zwischendialog ist es möglich (wie beim Standard-Anmeldedialog), das bisherige Passwort durch ein neues zu überschreiben.

Spezialfälle des Anmeldevorgangs für Terminals

Bei der Generierung von LTERM-Partnern mit automatischem KDCSIGN und der Anmeldung über Verteiler muss der Anmeldevorgang entsprechend angepasst werden.

LTERM-Partner mit automatischem KDCSIGN

Der Anmeldevorgang erhält beim Aufruf SIGN ST die Information, dass die Benutzerkennung bereits bekannt ist. Abhängig von der Generierung kann jetzt ein Zwischendialog zum Anfordern einer Ausweiskarte oder eines Passworts durchgeführt werden.

7.1.6.2 Anmeldevorgang für TS-Anwendungen

Beim Start des Anmeldevorgangs ist der Benutzer unter der Verbindungs-Benutzerkennung vorläufig angemeldet.

Im Anmeldevorgang können über den Aufruf SIGN ON die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn *open*UTM die Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmeldevorgangs unter der angegebenen Benutzerkennung angemeldet. Die Anmeldung wird abgelehnt, wenn die Berechtigungsdaten der TS-Anwendung nicht korrekt sind oder wenn unter der Verbindungs-Benutzerkennung ein Vorgang offen ist.

Ist die Anmeldung unter einer echten Benutzerkennung einmal fehlgeschlagen, dann muss in demselben Anmeldevorgang eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmeldevorgangs die Verbindung abgebaut. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Wird im Anmeldevorgang keine Benutzerkennung übergeben, dann wird der Benutzer beim ordnungsgemäßen Abschluss des Anmeldevorgangs endgültig unter der Verbindungs-Benutzerkennung angemeldet.

7.1.6.3 Anmeldevorgang für UPIC-Clients

Beim Anmelden über einen Anmeldevorgang sind zwei Fälle zu unterscheiden:

– Der UPIC-Client übergibt im UPIC-Protokoll Berechtigungsdaten an *open*UTM. Wenn *open*UTM die Daten akzeptiert, dann wird der Anmeldevorgang unter der übergebenen echten Benutzerkennung gestartet und der Client wird beim ordnungsgemäßen Abschluss des Anmeldevorgangs unter dieser Benutzerkennung angemeldet.

– Falls UPIC-Client im UPIC-Protokoll keine Berechtigungsdaten übergibt, dann wird der Anmeldevorgang unter der Verbindungs-Benutzerkennung gestartet. Im Anmeldevorgang können die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn *open*UTM diese Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmeldevorgangs unter dieser Benutzerkennung angemeldet. Werden keine Berechtigungsdaten übergeben, läuft die Conversation unter der Verbindungs-Benutzerkennung.

Ist die Anmeldung unter einer echten Benutzerkennung fehlgeschlagen, dann muss eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmeldevorgangs die Conversation beendet. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Damit Client-Programme unabhängig davon eingesetzt werden können, ob die UTM-Anwendung einen Anmeldevorgang verwendet oder nicht, können nicht gelesene Nachrichten vom Client bei Beendigung eines Teilprogramms des Anmeldevorgangs mit PEND PA, PEND PR, PEND PS oder PEND FC ohne vorhergehenden MPUT im Folgeteilprogramm gelesen werden.

7.1.6.4 Anwendungsmöglichkeiten für Anmeldevorgänge

Anmeldevorgänge bieten dem Anwender eine Reihe praktischer Nutzungsmöglichkeiten, die im Folgenden skizziert werden:

- TS-Anwendungen können sich über einen Anmeldevorgang mit einer echten Benutzerkennung an eine UTM-Anwendung anmelden. Dadurch werden sie in das Zugangs- und Zugriffskonzept von *open*UTM eingebunden.
- Der vom Benutzer eingegebene Name kann in eine Benutzerkennung umgesetzt werden, die per Generierung definiert ist (USER *username*).
- Im Falle eines DB/DC-übergreifenden Berechtigungskonzeptes kann man im 2. Teil des Anmeldevorgangs mit einem Datenbankaufruf das aktuelle Berechtigungsprofil für diesen USER aus der Datenbank holen und evtl. in einem benutzerspezifischen Langzeitspeicher (ULS) speichern.
- Der Anmeldevorgang kann den Benutzer im 2. Teil zur Änderung seines Passwortes auffordern, z.B. weil die Zeitspanne überwacht wird, in welcher der Benutzer dasselbe Passwort verwendet.
- Es kann eine Statistik über alle versuchten und erfolgreichen Anmeldungen erstellt werden.
- Auch im Falle eines nachfolgenden Vorgangswiederanlaufs kann der Anmeldevorgang dem Benutzer nützliche Informationen geben. Hierzu zählen Bulletin, Anzeige der Tastaturbelegung oder Anzeige des Vorgangswiederanlaufs. Dies erfordert natürlich einen zusätzlichen Dialogschritt.
- Wenn *open*UTM nach einem Aufruf SIGN OB (= KDCOFF BUT per Programm) den Anmeldevorgang startet, kann es sinnvoll sein, dass mit MGET die letzte Eingabe vom Terminal gelesen wird, wenn dort bereits neue Berechtigungsdaten eingetragen wurden.

7.1.6.5 Eigenschaften von Anmeldevorgängen

Ausgabe der letzten Dialognachricht durch den Anmeldevorgang

Liegt kein Vorgangswiederanlauf vor und wird der Anmeldevorgang mit MPUT PM und PEND FI beendet, dann wird die letzte Dialognachricht ausgegeben. Der Anwender kann dann nach Abschluss des Anmeldevorgangs mit dem gleichen Bildschirm weiterarbeiten, mit dem die letzte Sitzung beendet wurde - unabhängig davon, ob dies innerhalb oder außerhalb eines Vorgangs geschah.

Meldungen

Verwendet eine UTM-Anwendung einen Anmeldevorgang, werden die folgenden Meldungen nicht erzeugt (und demzufolge auch nicht an SYSLOG und MSGTAC ausgegeben):

K001, K002, K004 bis K008.

Die Meldung K033 (erfolgreiche Anmeldung) wird auch bei Verwendung eines Anmeldevorgangs ausgeben.

Fehlversuche beim Anmeldevorgang

Beim Anmeldevorgang können Fehlversuche des Benutzers beim Anmelden abgefangen werden: Akzeptiert *open*UTM die eingegebenen Berechtigungsdaten des Benutzers **nicht**, dann kann der Anmeldevorgang den Benutzer auffordern, die Eingabe zu wiederholen. Die maximale Anzahl der Eingabeversuche ist programmierbar. Wird sie überschritten, dann sollte sich der Anmeldevorgang beenden. Bei TS-Anwendungen und bei Terminals baut UTM die Verbindung ab, bei UPIC wird nur die Conversation beendet.

*open*UTM zählt außerdem alle Fehlversuche von einem Client mit, die in ununterbrochener Folge auftreten, auch über eine Folge von Anmeldevorgängen hinweg. Nach einer bei der Generierung festzulegenden Anzahl von fehlerhaften Anmeldeversuchen (siehe KDCDEF-Anweisung SIGNON, Operand SILENT-ALARM im *open*UTM-Handbuch „Anwendungen generieren“) meldet *open*UTM dieses Ereignis nach SYSLOG (stiller Alarm, Meldung K094). Anmeldeversuche unberechtigter Personen können aufgedeckt und mit einer MSGTAC-Routine abgewehrt werden.

Fehlverhalten des Anmeldevorgangs

*open*UTM kontrolliert, ob die Regeln für den Anmeldevorgang eingehalten werden. Das bietet auch einen Schutz gegen eventuelle Manipulationen der Teilprogramme des Anmeldevorgangs. Bei Fehlern dieser Art bricht *open*UTM den Anmeldevorgang mit PEND ER ab. Danach wird bei TS-Anwendungen und Terminals die Verbindung abgebaut; bei UPIC wird nur die Conversation beendet.

7.1.7 Verhalten bei gesperrten Clients/LTERM-Partnern

Verhalten bei gesperrten Clients

Clients können per Generierung (PTERM...,STATUS=OFF) oder per Administration gesperrt werden. Das Sperren eines Client hat folgende Wirkungen:

- Der Verbindungswunsch wird abgelehnt.
- Eine bestehende Verbindung bleibt erhalten; die Sperre wird erst dann wirksam, wenn von diesem Client ein neuer Verbindungswunsch kommt.

Verhalten bei gesperrten LTERM-Partnern

LTERM-Partner können per Generierung (LTERM...,STATUS=OFF) oder per Administration gesperrt werden.

Bei UPIC-Clients und TS-Anwendungen wirkt das Sperren des LTERM-Partners wie das Sperren des Clients.

Bei Terminals hat das Sperren eines LTERM-Partners folgende Wirkungen:

- Der Verbindungswunsch wird ausgeführt, aber nach dem Verbindungsaufbau wird die Meldung ausgegeben:

  ```
  K027 LTERM-Partner &LTERM gesperrt - Administrator verstaendigen
       oder KDCOFF eingeben.
  ```

- Eine bestehende Verbindung bleibt erhalten, die nächste Eingabe vom Terminal wird mit der Meldung K027 quittiert.

7.2 Anmeldeverfahren ohne Benutzerkennungen

Bei UTM-Anwendungen, für die keine Benutzerkennungen generiert sind, nimmt *open*UTM keine Berechtigungsprüfung vor. Die Clients werden unter ihrem LTERM-Namen bzw. Associations-Namen angemeldet. UPIC-Clients und OpenCPIC-Clients dürfen in diesem Fall keine echte Benutzerkennung übergeben.

Verwendet die UTM-Anwendung Anmeldevorgänge (Seite 98f), dann kann damit anschließend eine anwendungseigene Berechtigungsprüfung durchgeführt werden, zum Beispiel anhand einer Datenbank mit Berechtigungsdaten.

Wird kein Anmeldevorgang verwendet, dann kann der Benutzer nach erfolgreichem Verbindungsaufbau zu der UTM-Anwendung sofort mit dieser Anwendung arbeiten. Bei Terminals und TS-Anwendungen erhält der Benutzer von *open*UTM eine Ausgabe, die davon abhängt, ob für diesen LTERM-Partner noch ein offener Vorgang bekannt ist:

- Wenn für den LTERM-Partner in der Anwendung kein offener Vorgang bekannt ist, gibt *open*UTM folgende Meldung aus:

  ```
  K001 Verbunden mit Anwendung beispiel – Bitte Eingabe
  ```

  Bei Terminals wird, falls generiert, das Startformat zu diesem LTERM-Partner ausgegeben. Der Benutzer kann dann Vorgänge starten und UTM-Benutzer-Kommandos eingeben.

- Wenn für diesen LTERM-Partner in der Anwendung noch ein offener Vorgang bekannt ist, so bekommt der Benutzer die Ausgabe vom letzten Sicherungspunkt seines unterbrochenen Vorgangs auf den Bildschirm, und er kann den Vorgang fortsetzen. Sehen Sie dazu auch *open*UTM-Handbuch „Anwendungen programmieren mit KDCS“. Dafür ist unter anderem Voraussetzung, dass für diesen LTERM-Partner RESTART=YES generiert wurde. Das heißt aber auch, dass der Benutzer eventuell den Vorgang eines anderen Benutzers fortsetzen kann.

Beachten Sie, dass *open*UTM in einer Anwendung ohne Benutzerkennungen einen Vorgang an den LTERM-Partner koppelt. Ein unterbrochener Vorgang kann deshalb auch nur vom selben Client fortgesetzt werden. Es sei denn, die Zuordnung LTERM-Partner und physikalischer Client (festgelegt in der PTERM-Anweisung) wird per Administration, z.B. mit dem Administrationskommando KDCSWTCH, geändert.

Das Verhalten bei gesperrten Clients ist das gleiche wie mit Benutzerkennungen, siehe Seite 103.

ACHTUNG!

In einer UTM-Anwendung ohne Benutzerkennungen hat jeder Benutzer die Administrationsberechtigung.

7.3 UTM-Services aufrufen

Ist die UTM-Berechtigungsprüfung erfolgreich verlaufen, so ist der Benutzer berechtigt, mit der UTM-Anwendung zu arbeiten, d.h er kann neue Vorgänge (=Services) starten (siehe unten) oder offene Vorgänge fortsetzen.

Die Abschnitte „Vorgänge vom Terminal aus starten“ auf Seite 105 bis „Vorgänge von TS-Anwendungen aus starten“ auf Seite 107 zeigen, wie bei den einzelnen Client-Typen neue Vorgänge gestartet werden. Der Abschnitt „Vorgangswiederanlauf“ auf Seite 108 beschreibt die Situation, wenn für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist.

7.3.1 Vorgänge vom Terminal aus starten

Nach erfolgreicher Anmeldung kann der Benutzer einen Vorgang starten, indem er einen Transaktionscode (TAC) eingibt oder eine entsprechend generierte Funktionstaste drückt.

Starten eines Vorgangs durch Eingabe eines Transaktionscodes

Wenn kein Anmeldevorgang durchlaufen wird, kann der Benutzer folgende Situationen vorfinden:

– Im Zeilenmodus gibt *open*UTM die folgende Meldung aus:

```
K008 Anmeldung akzeptiert - Bitte Eingabe
```

Der Benutzer kann einen Vorgang starten, indem er einen TAC und eventuell eine Nachricht eingibt. Die ersten acht Zeichen seiner Eingabe werden von *open*UTM als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, dann muss er durch ein Leerzeichen von der Nachricht getrennt sein.

– Wenn für den Benutzer ein Startformat generiert wurde, dann muss er die entsprechenden Felder des Startformats ausfüllen. Das Feld mit dem TAC, auch Steuerfeld genannt, steht dabei nicht notwendigerweise an erster Stelle.

Wenn ein Anmeldevorgang durchlaufen wird, dann bestimmt der Anmeldevorgang den nächsten Schritt. Der Benutzer erhält dann eine Ausgabe im Zeilenmodus oder es wird direkt ein Vorgang gestartet.

Starten eines Vorgangs über eine Funktionstaste

Ein TAC kann per Generierung auf eine Funktionstaste gelegt werden, z.B. F10, siehe KDCDEF-Anweisung SFUNC. Wenn der Benutzer diese Taste betätigt, dann wird der Vorgang gestartet, unabhängig davon, ob ein Benutzer im Zeilenmodus arbeitet oder ein Startformat vor sich hat.

Tastaturbelegung bei UNIX-Terminals

Im Zeilemodus gilt folgende Tastaturbelegung:

Taste	Wirkung
Korrektur	Wie in der UNIX-Shell
return	Die Daten werden an die Anwendung gesendet.
END DEL	Der Dialog wird beendet und der Benutzer von der Anwendung abgemeldet. Die Taste DEL wirkt **nicht**, wenn sich der Benutzer mit dem Schalter -D angemeldet hat.
andere Systemtasten	Es werden sowohl die Escape-Sequenzen der Taste als auch die Daten an die Anwendung gesendet, daher auf **keinen** Fall verwenden.

Tastaturbelegung in Windows

Wenn man in Windows an der Console über die DOS-Eingabeaufforderung mit der Anwendung arbeitet, gilt folgende Tastaturbelegung:

Taste	Wirkung
Korrektur	Wie in der DOS-Shell
return	Die Daten werden an die Anwendung gesendet.
END CTRL+C	Der Dialog wird beendet und der Benutzer von der Anwendung abgemeldet.
DEL	Die Taste wird ignoriert
Cursor-Tasten	Wie in der DOS-Eingabeaufforderung, d.h. Historie der letzten Eingaben
andere Systemtasten	Es werden sowohl die Escape-Sequenzen der Taste als auch die Daten an die Anwendung gesendet, daher auf **keinen** Fall verwenden.

Eingabe ungültiger Transaktionscodes

Wenn der Benutzer einen falschen TAC eingibt, dann erhält er die Meldung

```
K009  Der Transactionscode <tac> ist ungültig – Bitte Eingabe
```

Falls in der Anwendung ein Dialogvorgang BADTACS generiert ist, dann wird stattdessen der Vorgang BADTACS gestartet. Nachdem der Dialogvorgang BADTACS beendet wurde, bleibt der Benutzer angemeldet und kann wie oben beschrieben einen Vorgang starten.

7.3.2 Vorgänge vom UPIC-Client und OSI-TP-Client aus starten

Nach dem Verbindungsaufbau können die OSI-TP- oder UPIC-Clients Conversations starten. Dazu wird der TAC durch den Client gesetzt, z.B. über die Funktion *Set_TP_Name* der CPI-C-Schnittstelle oder durch einen entsprechenden Eintrag in der Side Information Datei. Dieser TAC wird an *open*UTM übergeben, eventuell zusammen mit Berechtigungsdaten. Nach erfolgreicher Berechtigungsprüfung gilt:

– Bei OpenCPIC-Clients und bei UPIC-Clients ohne Anmeldevorgang wird der zum TAC gehörige Vorgang sofort gestartet.

– Bei UPIC-Clients mit Anmeldevorgang wird der zum TAC gehörige Vorgang erst nach dem Ende des Anmeldevorgangs gestartet.

Am Ende der Conversation wird der Benutzer wieder abgemeldet, wenn er sich für diese Conversation unter einer echten Benutzerkennung angemeldet hat.

7.3.3 Vorgänge von TS-Anwendungen aus starten

TS-Anwendungen verhalten sich ähnlich wie Terminals:

– Ohne Anmeldevorgang erhält die TS-Anwendung die Meldung K001, falls dieser Meldung das Meldungsziel PARTNER zugewiesen wurde, siehe Beschreibung des Tools KDCMMOD im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

  Anschließend kann die TS-Anwendungen einen Vorgang starten, indem sie einen TAC und eventuell eine Nachricht an die UTM-Anwendung übergibt. Dazu werden die ersten 8 Zeichen der Nachricht als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, muss er durch Leerzeichen von der Nachricht getrennt sein.

– Wird ein Anmeldevorgang durchlaufen, dann bestimmt dieser den nächsten Schritt. Der Anmeldevorgang kann entweder direkt einen Vorgang starten oder eine Ausgabenachricht an die TS-Anwendung schicken. Wird eine Ausgabenachricht an die TS-Anwendung geschickt, dann muss die nächste Nachricht in den ersten 8 Zeichen einen TAC enthalten, d.h. es gilt das gleiche wie ohne Anmeldevorgang, siehe oben.

Nach dem Beenden des Vorgangs wird die TS-Anwendung von UTM abgemeldet.

7.3.4 Vorgangswiederanlauf

Wenn sich ein Client unter einer bestimmten Benutzerkennung anmeldet, die mit RESTART=YES generiert ist, und wenn für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist, dann wird in der Regel ein Vorgangswiederanlauf durchgeführt. Beim Vorgangswiederanlauf sendet *open*UTM die Ausgabe vom letzten Sicherungspunkt des unterbrochenen Vorgangs an den Client, der Benutzer kann den Vorgang anschließend fortsetzen.

Abhängig von der Art des Client und vom Anmeldeverfahren gilt für den Vorgangswiederanlauf Folgendes:

– Standard-Anmeldeverfahren für Terminals und TS-Anwendungen:
*open*UTM führt den Vorgangswiederanlauf automatisch durch.

– Standard-Anmeldeverfahren für UPIC-Clients und OSI-TP-Clients:
Der Client muss eine spezielle Conversation starten, die mittels des UTM-Benutzerkommandos KDCDISP den Wiederanlauf anfordert (siehe z.B. Handbuch „*open*UTM-Client fürTrägersystem UPIC“). Über OSI-TP-Clients ist kein Fortsetzen des Vorgangs möglich, wenn die Functional Unit „Commit“ ausgewählt wurde.

– Anmelden über Anmeldevorgang:
Der Anmeldevorgang muss den Wiederanlauf initiieren oder den offenen Vorgang abnormal beenden.

In einer Anwendung mit Benutzerkennungen ist ein Vorgang an die Benutzerkennung gebunden. Deshalb kann der Benutzer einen unterbrochenen Vorgang auch an einem anderen Client fortsetzen, sofern der LTERM-Partner des Client die Berechtigung dazu hat und der Typ des Clients gleich bleibt.

7.4 Berechtigungskonzept von *open*UTM

*open*UTM bietet zusätzlich zum Zugangsschutz durch Benutzerkennungen ein ausgefeiltes Zugangs- und Zugriffskonzept. Damit lässt sich steuern, welcher Benutzer über welche LTERM-Partner auf welche Services der UTM-Anwendung zugreifen darf.

Dabei gibt es die benutzerorientierte Variante (**Lock-/Keycode**-Konzept) und die rollenorientierte Variante (**Access-List-**Konzept). Diese Varianten werden mit Hilfe von Lockcodes, Access-Lists, Keysets und Keycodes generiert:

– Ein Service wird entweder mit Lockcodes (Lock-/Keycode-Konzept) oder mit einer Access-List (Access-List-Konzept) geschützt (TAC-Anweisung, LOCK= bzw. ACCESS-LIST=).

– Eine Benutzerkennung erhält ein Keyset mit einem oder mehreren Keycodes (USER-Anweisung, KSET=). Die Keycodes definieren die Berechtigungen.

– Ein LTERM-Partner erhält ein Keyset mit einem oder mehreren Keycodes sowie Lockcodes, wenn das Lock-/Keycode-Konzept verwendet wird (LTERM- bzw. TPOOL-Anweisung, Operanden KSET= und LOCK=).

– Keysets werden eigens in KSET-Anweisungen definiert

Die folgende Tabelle zeigt für die beiden Konzept-Varianten, unter welchen Voraussetzungen ein Benutzer sich anmelden kann und wann er einen Vorgang starten darf oder fortsetzen darf (nach einem Vorgangswiederanlauf).

Tätigkeit	**Voraussetzungen**	
	Lock-/Keycode-Konzept	**Access-List-Konzept**
Anmelden über bestimmten LTERM-Partner	Ein Keycode der Benutzerkennung stimmt mit dem Lockcode des LTERM-Partners überein.	Anmelden immer möglich.
Starten eines Vorgangs	Benutzerkennung und LTERM-Partner besitzen einen Keycode, der mit dem Lockcode des TACs übereinstimmt.	Benutzerkennung und LTERM-Partner besitzen jeweils einen Keycode, der in der Access-List des TACs enthalten ist. Die Keycodes von Benutzerkennung und LTERM müssen nicht identisch sein.
Vorgang fortsetzen (nach Vorgangswiederanlauf)	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss mit dem Lockcode des Folge-TACs übereinstimmen.	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss in der Access-List des Folge-TACs enthalten sein.

Meldungen bei fehlenden Berechtigungen

Wenn Berechtigungen fehlen, kann der Terminalbenutzer folgende Meldungen erhalten (beim Anmeldevorgang kommt ein entsprechender Returncode):

```
K005 Die Benutzerkennung user ist gesperrt
```

wenn der Keycode des Benutzers nicht mit dem Keycode des LTERM-Partners übereinstimmt (Anmeldevorgang: Returncode U02).

```
K009 Der Transaktionscode ist ungueltig
```

wenn der Benutzer oder das LTERM nicht die Berechtigung haben, den Vorgang zu starten. Wenn ein BADTAC-Vorgang generiert ist, dann wird stattdessen der BADTAC-Vorgang gestartet.

```
K123 LTERM hat nicht die Berechtigung den Vorgang fortzusetzen - Bitte Anmelden
```

wenn der LTERM-Partner, über den sich der Benutzer beim Vorgangswiederanlauf angemeldet hat, nicht die Berechtigung hat, den Folge-TAC zu starten (Anmeldevorgang: Returncode U16). Diese Meldung kann insbesondere dann auftreten, wenn ein Benutzer den Vorgang von einem anderen Terminal und damit einem anderen LTERM fortsetzt.

Weitere Informationen finden Sie im *open*UTM-Handbuch „Konzepte und Funktionen“ sowie im *open*UTM-Handbuch „Anwendungen generieren“.

7.5 Abmelden von der UTM-Anwendung

Die folgenden Abschnitte beschreiben die verschiedenen Möglichkeiten, wie sich ein Client bei der UTM-Anwendung abmelden kann oder ein Client durch UTM abgemeldet wird. Dabei unterscheiden sich Terminals von allen anderen Clients, denn nur von Terminals aus können sich Benutzer explizit von der Anwendung abmelden.

Abmelden bei Zeitüberschreitung - Zeitüberwachung

Bei der Generierung können maximale Wartezeiten festegelegt werden durch:

– die Operanden TERMWAIT= (PEND KP-Timer) und PGWTTIME= (PGWT-Timer) in der KDCDEF-Steueranweisung MAX

– den Operanden IDLETIME= (Transaktionsende-Timer) der PTERM-Anweisung bzw. der OSI-LPAP-Anweisung (bei OSI-TP-Clients).

Wenn eine mit diesen Timern eingestellte Wartezeit abgelaufen ist, dann wird bei Terminals folgende Meldung ausgegeben:

```
K021 Eine Eingabe ist nicht in der vorgegebenen Zeit erfolgt
```

Anschließend meldet *open*UTM die Benutzerkennung ab und baut die Verbindung zum Client ab. Der Client kann sich danach wieder bei der Anwendung anmelden und den Vorgang fortsetzen, siehe Abschnitt „Vorgangswiederanlauf“ auf Seite 108.

Abmelden mit KDCOFF-Kommando

Der Terminal-Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF bzw. KDCOFF BUT von der UTM-Anwendung abmelden. Sehen Sie hierzu auch das UTM-Benutzerkommando KDCOFF auf Seite 117.

KDCOFF aus einem Programm

*open*UTM bietet die Funktionsaufrufe SIGN OF und SIGN OB, mit denen in einem Dialog-Teilprogramm die Wirkung des Benutzerkommandos KDCOFF bzw. KDCOFF BUT ausgelöst werden kann. SIGN OF/OB ist für Terminals, UPIC-Clients und TS-Anwendungen möglich. In Teilprogrammen, die für einen OSI-TP-Client laufen, sind diese Aufrufe nicht erlaubt.

SIGN OF und SIGN OB wirken wie folgt:

SIGN-Aufruf	Kommando	Auswirkung
SIGN OF	KDCOFF	*open*UTM baut Verbindung zum Client ab.
SIGN OB	KDCOFF BUT	Für Terminals bleibt die Verbindung bleibt erhalten, der Benutzer wird abgemeldet. Bei UPIC-Clients und TS-Anwendungen wird die Verbindung abgebaut (wie SIGN OF).

Der Aufruf wirkt bei Terminals und UPIC-Clients/TS-Anwendungen unterschiedlich:

- Bei Terminals gibt *open*UTM zunächst die MPUT-Nachricht und die Meldung K095 an das Terminal aus. Erst durch die nächste (beliebige) Eingabe vom Terminal wird der Benutzer abgemeldet und (bei SIGN OF) die Verbindung abgebaut.
- Bei UPIC-Clients und TS-Anwendungen wird die MPUT-Nachricht gesendet und anschließend sofort der Verbindungsabbau initiiert.

Im Folgenden werden einige Anwendungsmöglichkeiten für den Funktionsaufruf SIGN OF/OB skizziert:

- Anwendungen mit besonderen Sicherheitsanforderungen. Nach dem Anmelden darf ein Benutzer nur einen einzigen Vorgang bearbeiten.
- Der Steuerteil des Bildschirms bietet als mögliche nächste Aktionen auch „Abmelden“ oder „Neu-Anmeldung“ an. Das Folgeteilprogramm erzeugt dann abhängig von der Eingabe einen Aufruf SIGN OF bzw. SIGN OB. Nach der Dialogausgabe dieses Teilprogramms und der darauf folgenden Eingabe wird entweder die Verbindung zum Terminal abgebaut oder aber der Anmeldevorgang gestartet.

7.6 UTM-Benutzerkommandos für Terminals

Dieser Abschnitt enthält eine Beschreibung aller UTM-Benutzerkommandos, die dem Terminal-Benutzer zur Verfügung stehen:

– KDCOUT, um asynchrone Nachrichten anzufordern

– KDCDISP, um die letzte Dialognachricht nochmals anzufordern

– KDCLAST, um die letzte Ausgabe zu wiederholen

– KDCOFF, um sich abzumelden

UTM-Benutzerkommandos können bei der Generierung der Anwendung mit der KDCDEF-Anweisung SFUNC einer K-/F-Taste zugeordnet werden, und können dann über Funktionstaste eingegeben werden.

KDCOUT - Asynchrone Nachricht ausgeben

Mit dem KDCOUT-Kommando kann der Benutzer die Ausgabe asynchroner Nachrichten anfordern.

*open*UTM kündigt asynchrone Nachrichten mit der folgenden Meldung an:

```
K012 nnn Nachricht(en) vorhanden
```

Die Meldung erscheint zusammen mit der nächsten Dialogausgabe an dieses Terminal in der Systemzeile. Mit *nnn* wird die Anzahl der asynchronen Nachrichten angegeben. Der Benutzer kann diese Nachrichten mit dem Kommando KDCOUT abholen. Sind jedoch keine Nachrichten für das Terminal vorhanden, so meldet *open*UTM:

```
K020 keine Nachricht vorhanden
```

Wird eine asynchrone Nachricht mit KDCOUT abgeholt, dann wird sie durch die nächste Eingabe gelöscht, außer bei Eingabe von KDCLAST (siehe Seite 116).

Die KDCDEF-Anweisung LTERM ..., RESTART= NO hat zur Folge, dass beim Verbindungsaufbau oder -abbau zu diesem LTERM-Partner anstehende asynchrone Nachrichten gelöscht werden.

Die Funktionsvarianten von *open*UTM haben folgende Auswirkungen auf die Behandlung von asynchronen Nachrichten:

- Bei UTM-S-Anwendungen bleiben asynchrone Nachrichten auch über Unterbrechungen des Anwendungslaufes hinweg gesichert, bis sie mit KDCOUT abgeholt werden.
- Bei UTM-F-Anwendungen werden asynchrone Nachrichten nur während des Anwendungslaufes gespeichert. Sie gehen mit der Beendigung des Anwendungslaufes verloren.

KDCDISP - Letzte Dialognachricht ausgeben

In einer laufenden UTM-Anwendung kann sich der Benutzer mit dem KDCDISP-Kommando die letzte Dialognachricht noch einmal ausgeben lassen.

Nach Beendigung und erneutem Start der Anwendung kann der Benutzer sich bei der Variante UTM-S nach dem Anmelden mit dem KDCDISP-Kommando die Dialogausgabenachricht vom letzten Sicherungspunkt noch einmal ausgeben lassen.

Gibt der Benutzer das KDCDISP-Kommando nach Abschluss des Anmeldevorgangs oder nach der Rückkehr von einem eingeschobenen Vorgang ein, dann zeigt *open*UTM nochmals den letzten Bildschirm der letzten Session, bzw. den letzten Bildschirm des unterbrochenen Vorgangs.

Das KDCDISP-Kommando ist in folgenden Situationen nützlich:

- Aufgrund von Fehlbedienung am Terminal ist der Bildschirminhalt nach einer Dialogausgabe teilweise oder vollständig zerstört.
- Der Benutzer hat während der Bearbeitung eines Vorgangs asynchrone Nachrichten am Bildschirm erhalten. Die asynchronen Nachrichten wurden entweder durch *open*UTM direkt gesendet oder der Benutzer hat sie mit KDCOUT selbst angefordert. Wenn der Benutzer danach den offenen Vorgang fortsetzen möchte, lässt er sich dazu mit einem KDCDISP-Kommando die letzte Dialogausgabe nochmals ausgeben.
- Nach Beendigung und erneutem Start der UTM-Anwendung kann der Benutzer (zur Orientierung) mit dem KDCDISP-Kommando die letzte Dialogausgabe des vor Beendigung der Anwendung abgeschlossenen Vorgangs wiederholen. Das gilt aber nur, wenn es sich um eine UTM-S-Anwendung handelt und wenn der Vorgangswiederanlauf nicht explizit ausgeschaltet wurde durch die KDCDEF-Anweisung USER ...,RESTART=NO (bzw. LTERM ...,RESTART=NO, wenn die Anwendung ohne Benutzerkennungen generiert wurde).

KDCLAST - Letzte Ausgabe wiederholen

Das Kommando KDCLAST ermöglicht die Wiederholung der letzten Ausgabenachricht am Terminal unabhängig davon, ob es eine Dialognachricht oder eine asynchrone Nachricht war.

Wurde als letztes eine asynchrone Nachricht ausgegeben, so wird diese Ausgabe mit KDCLAST wiederholt. Die asynchrone Nachricht wird dabei aber noch nicht freigegeben.

Wird das Kommando KDCLAST nach Abschluss des Anmeldevorgangs eingegeben, dann zeigt *open*UTM nochmals den letzten Bildschirm des Anmeldevorgangs. Wird es nach der Rückkehr von einem eingeschobenen Vorgang eingegeben, wird der letzte Bildschirm des eingeschobenen Vorgangs ausgegeben.

KDCOFF - Abmelden von einer UTM-Anwendung

Der Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF von der UTM-Anwendung abmelden. Dadurch wird die Verbindung zwischen Dialogterminalprozess und UTM-Anwendung abgebaut und der Dialogterminalprozess beendet.
Wurde der Dialogterminalprozess nach erfolgreichem Anmelden automatisch durch UNIX gestartet (siehe Seite 87), so wird auch der Dialog mit UNIX beendet.

Meldet er sich während der Bearbeitung eines Vorgangs am Transaktionsende ab, wird die Bearbeitung unterbrochen. Sie kann nach späterem Anmelden an die UTM-Anwendung wieder fortgesetzt werden.

KDCOFF BUT

Durch Eingabe von KDCOFF BUT kann der Benutzer sich so abmelden, dass die Verbindung zwischen Dialogterminalprozess und UTM-Anwendung bestehen bleibt. Er wird gleich zum erneuten Anmelden aufgefordert, bzw. der Anmeldevorgang wird gestartet.

Meldungen

Nach Eingabe von KDCOFF [BUT] reagiert *open*UTM mit einer der Meldungen:

`K019 KDCOFF von Anwendung beispiel akzeptiert`

Der Benutzer hat KDCOFF eingegeben, oder er hat in einer Anwendung ohne Benutzerkennungen KDCOFF BUT eingegeben. Das Terminal ist nicht mehr mit der UTM-Anwendung verbunden.

`K018 KDCOFF von Anwendung beispiel akzeptiert - Bitte Anmelden`

In einer Anwendung mit Benutzerkennungen und ohne Anmeldevorgang hat der Benutzer KDCOFF BUT eingegeben. *open*UTM fordert ihn zum erneuten Anmelden mit einer Benutzerkennung auf. Dies gilt auch dann, wenn der Benutzer sich ohne Schalter -S angemeldet hat.

`K003 Das Kommando KDCOFF ist in dieser Situation nicht erlaubt`

Die Eingabe erfolgte nach einem PEND KP-Aufruf oder blockierenden Aufruf (z.B. PGWT) des Teilprogramms.

8 Programmaustausch im Betrieb

Für den Austausch von Programmen im laufenden Betrieb bietet *open*UTM zwei unterschiedliche Verfahren an:

- Austausch des gesamten Anwendungsprogramms mit Hilfe des Tools KDCPROG und des Administrationskommandos KDCAPPL bzw. eines Administrationsprogramms, das KDCADMI mit Operationscode KC_CHANGE_APPLICATION aufruft.
- Austausch von Shared Objects, d.h. von Teilen des Anwendungsprogramms mit Hilfe des Administrationskommandos KDCPROG bzw. eines Administrationsprogramms, das KDCADMI mit Operationscode KC_MODIFY_OBJECT und Objekttyp KC_LOAD_MODULE aufruft.

Die Verfahren können gemischt werden, d.h. eine Anwendung, die Shared Objects enthält, kann auch als Ganzes ausgetauscht werden. Beide Verfahren werden im Folgenden beschrieben.

8.1 Anwendung austauschen

*open*UTM bietet Ihnen die Möglichkeit, das Anwendungsprogramm während des Anwendungslaufs auszutauschen. D.h. Sie können z.B. Teilprogramme Ihres Anwendungsprogramms verändern, eine neue Version des Anwendungsprogramms erstellen und diese Version des Anwendungsprogramms in Betrieb nehmen, ohne dass Sie den Anwendungslauf beenden müssen.

Die auszutauschende Anwendung kann sowohl nur aus statisch in *utmwork* eingebundenen Teilprogrammen bestehen als auch Teilprogramme enthalten, die als Shared Objects gebunden sind.

Bei den Änderungen des Anwendungsprogramms ist Folgendes zu beachten:

- In Anwendungen **ohne** Shared Objects dürfen die Änderungen keine Auswirkung auf die KDCDEF-Generierung und die KDCFILE haben. D.h. die Funktion „Anwendungsaustausch" können Sie **nicht** anwenden, wenn Sie das Anwendungsprogramm um neue Teilprogramme erweitern wollen. Werden neue Teilprogramme hinzugefügt, dann sind diese nicht in den Tabellen der KDCFILE enthalten.

- In Anwendungen **mit** Shared Objects können Sie neue Teilprogramme ins Anwendungsprogramm aufnehmen. Diese Teilprogramme müssen jedoch in Shared Objects gebunden werden, die in der Konfiguration der Anwendung enthalten sind. Die Teilprogramme und die zugehörigen Transaktionscodes müssen per Administration in die Tabellen der KDCFILE eingetragen werden.

Fehlen nach dem Austausch Teilprogramme, die in dem vorherigen Anwendungsprogramm vorhanden waren, so können Aufträge für TACs bei der Anwendung eintreffen, für die kein Teilprogramm mehr vorhanden ist. Diese Aufträge beendet *open*UTM mit PEND ER. Die Transaktionscodes können vom UTM-Administrator aus der Konfiguration gelöscht werden.

Die verschiedenen Versionen Ihres Anwendungsprogramms und den Austausch verwaltet das Tool KDCPROG (siehe Abschnitt „Das Tool KDCPROG“ auf Seite 127). KDCPROG verwaltet die Versionen des Anwendungsprogramms in einem Dateigenerationsverzeichnis (FGG=**F**ile **G**eneration **G**roup).

Wird ein als Shared Object vorliegendes Teilprogramm ausgetauscht, so wird zunächst das alte Shared Object durch das neue ersetzt. Das führt dazu, dass die Verweise vom statischen Teil des Anwendungsprogramms auf das Shared Object dann unbefriedigt sind. Erst wenn der UTM-Administrator mit dem Administrationskommando KDCAPPL PROG=NEW | OLD den Programmaustausch für das Anwendungsprogramm initiiert, werden alle unbefriedigten Verweise auf das Shared Object befriedigt und das neue Shared Object im Anwendungsprogramm wirksam.

8.1.1 Voraussetzungen für den Anwendungsaustausch

Folgende Arbeitsschritte müssen Sie für den Anwendungsaustausch durchführen:

1. **Dateigenerationsverzeichnis PROG einrichten**

   Damit Sie ein UTM-Anwendungsprogramm im laufenden Betrieb austauschen können, sollten Sie die verschiedenen Versionen des Anwendungsprogramms (auch das aktuell geladene) mit Hilfe des Tools KDCPROG verwalten. Dazu müssen Sie mit KDCPROG im Basisverzeichnis *filebase* der Anwendung das Dateigenerationsverzeichnis PROG anlegen (Funktion KDCPROG CREATE). Falls Sie kein Dateigenerationsverzeichnis erstellt haben, lädt KDCAPPL PROG= (bzw. der entsprechende Aufruf an der Programmschnittstelle zur Administration) das Anwendungsprogramm *utmwork* aus dem Basisverzeichnis *filebase* neu.

   In dem Dateigenerationsverzeichnis (im Folgenden FGG genannt; **F**ile **G**eneration **G**roup) verwaltet KDCPROG die verschiedenen Versionen Ihres Anwendungsprogramms. Sehen Sie dazu Abschnitt „Dateigenerationsverzeichnis PROG“ auf Seite 123. Dazu müssen Sie jedes *utmwork*-Programm zusammen mit der zugehörigen Symboltabelle *nmutmwork* als eine Generation in der FGG ablegen.

Die FGG müssen Sie nur einmal einrichten. Sie bleibt erhalten. In ihr können Sie mit Hilfe von KDCPROG mehrere Versionen Ihres Anwendungsprogramms verwalten.

Die FGG können Sie einrichten, bevor oder nachdem Sie die erste Version des Anwendungsprogramms erstellt haben. Das Anwendungsprogramm, d.h. *utmwork* und die zugehörige Symboltabelle *nmutmwork*, erzeugen Sie wie im Kapitel „Anwendungsprogramm erzeugen“ auf Seite 13 beschrieben.

ACHTUNG!

Für jede Anwendung kann nur eine FGG für den Anwendungsaustausch existieren. Eine bereits vorhandene FGG und die darin enthaltenen Versionen des Anwendungsprogramms werden gelöscht, wenn mit KDCPROG CREATE eine neue FGG eingerichtet wird.

2. Anwendungsprogramm in die FGG transferieren

Sie transferieren Ihr Anwendungsprogramm mit KDCPROG TRANSFER in die FGG PROG (sehen Sie dazu auch Abschnitt „TRANSFER - utmwork in die FGG übertragen“ auf Seite 129). Sie müssen dem Anwendungsprogramm dabei die Version - d.h. die absolute Generationsnummer - 0001 zuordnen.

Zusätzlich muss mit KDCPROG SWITCH die Basis der FGG auf die Generationsnummer 0001 umgeschaltet werden.

Danach können Sie die Anwendung starten, wie im Kapitel „UTM-Anwendung starten“ auf Seite 53 beschrieben. *open*UTM lädt *utmwork* aus der FGG.

3. Weitere Versionen des Anwendungsprogramms erstellen und in die FGG transferieren

Unabhängig davon, ob Ihre Anwendung gestartet ist oder nicht, können Sie weitere Versionen Ihres Anwendungsprogramms erstellen. *utmwork* und die zugehörige Symboltabelle *nmutmwork* transferieren Sie mit KDCPROG TRANSFER in die FGG. Dabei ordnen Sie jeder Version des Anwendungsprogramms eine Generationsnummer zu. Die Generationsnummern in der FGG müssen fortlaufend ansteigend sein.

ACHTUNG!

Existiert in der FGG bereits eine Generation des Anwendungsprogramms mit der Generationsnummer, die Sie beim Transfer einer neuen Generation angeben, so wird dieses ohne Warnung überschrieben.

Wenn diese Voraussetzungen erfüllt sind, können Sie den Anwendungsaustausch jederzeit und beliebig oft anfordern. Dazu stehen Ihnen folgende Möglichkeiten zur Verfügung:

das Administrationskommando KDCAPPL PROG=... und an der Programmschnittstelle zur Administration der KDCADMI-Operationscode KC_CHANGE_APPLICATION
Beide Möglichkeiten sind im *open*UTM-Handbuch „Anwendungen administrieren“ beschrieben.

Wenn im Folgenden auf Aktionen hingewiesen wird, die mit KDCAPPL PROG= durchgeführt werden können, dann gilt dies auch für Administrationsprogramme, die KDCADMI mit Operationscode KC_CHANGE_APPLICATION absetzen.

Beim Start lädt *open*UTM die Version des Anwendungsprogramms, dessen Generationsnummer Basisnummer der FGG ist. Sehen Sie dazu Abschnitt „Dateigenerationsverzeichnis PROG“ auf Seite 123.

Bei einem Programmaustausch während des Anwendungslaufs setzt *open*UTM die Basis auf die aktuell geladene Version. So ist gewährleistet, dass der nächste Start mit der zuletzt geladenen Version des Anwendungsprogramms erfolgt.

8.1.2 Dateigenerationsverzeichnis PROG

*open*UTM verwaltet die verschiedenen Versionen Ihres Anwendungsprogramms, die Sie für den Austausch verwenden, in dem Dateigenerationsverzeichnis (FGG) PROG. Die FGG richten Sie wie folgt ein (siehe dazu auch Abschnitt „Das Tool KDCPROG“ auf Seite 127):

Unter UNIX mit dem Shell-Kommando:

utm-dateiverzeichnis`/ex/kdcprog CREATE` *operanden*

Unter Windows in der DOS-Eingabeaufforderung mit dem Kommando:

utm-dateiverzeichnis`\ex\kdcprog CREATE` *operanden*

In der folgenden Tabelle sind den zentralen FGG-Begriffen jeweils die *open*UTM-spezifischen Definitionen bezüglich Versionsverwaltung und Austausch des Anwendungsprogramms gegenübergestellt.

FGG-Begriffe	**Versionsverwaltung für den Anwendungsaustausch durch *open*UTM**
Generation des FGG	Eine Version Ihres Anwendungsprogramms besteht aus *utmwork* und der zugehörigen Symboltabelle *nmutmwork*. Jedes Paar *utmwork*, *nmutmwork* bildet eine *Generation* der FGG.
absolute Generationsnummer	Beim Transfer einer Generation in die FGG ordnen Sie der Generation eine laufende Nummer zu. Dies ist die absolute Generationsnummer der Generation. *open*UTM spricht die einzelnen Generationen über ihre Generationsnummern an.
Name der Generation	Der Name einer Generation ist: – unter UNIX: PROG/*absolute_generationsnummer* – unter Windows: PROG*absolute_generationsnummer* Dabei ist: 0001 ≤ *absolute_generationsnummer* ≤ 9999. Dem ersten Anwendungsprogramm, das Sie in die FGG transferieren, müssen Sie die absolute Generationsnummer 0001 zuordnen. Der Name der Generation ist damit PROG/0001 bzw. PROG\001.
aufsteigende Generationsnummer	Die Generationsnummern der folgenden Generationen müssen fortlaufend aufsteigend sein (z.B. 0002, 0003, 0004, usw.).

FGG-Begriffe	Versionsverwaltung für den Anwendungsaustausch durch *open*UTM
Basis der FGG	Eine Generation der FGG ist die Basis der FGG. UTM lädt beim Start der Anwendung immer die Basis.
Basisnummer	Die absolute Generationsnummer der Basis ist die Basisnummer der FGG. Nach dem Einrichten der FGG ist 0001 die Basisnummer und die erste in die FGG transferierte Generation die Basis der FGG. Während des Anwendungslaufs schaltet *open*UTM die Basis immer auf die aktuell geladene Generation um, d.h. bei einem Programmaustausch schaltet *open*UTM die Basis auf die beim Austausch geladene Generation um.
relative Generationsnummer	Jeder Generation einer FGG wird relativ zur Basis eine relative Generationsnummer und ein relativer FGG-Name zugeordnet.
relativer FGG-Name	Der relative FGG-Name einer Generation ist PROG(*relative_Generationsnr.*) wobei: *relative_generationsnr.* = *absolute_generationsnr.* - *Basisnummer.* Die relative Generationsnummer wird immer mit Vorzeichen (+ / -) angegeben.
maximale Anzahl der Generationen	Beim Einrichten der FGG legen Sie die maximale Anzahl der Generationen fest, die gleichzeitig in der FGG existieren dürfen.
erste Generation	Ist die maximale Anzahl der Generationen erreicht, dann geht *open*UTM beim nächsten Transfer wie folgt vor: – *open*UTM erzeugt eine neue Generation mit einer neuen Generationsnummer – *open*UTM löscht die Generation der FGG mit der niedrigsten Generationsnummer, d.h. die erste Generation.
letzte Generation	Die Generation der FGG mit der höchsten Generationsnummmer ist die letzte Generation der FGG.

Beispiele

1. Beim Start der Anwendung ist die Generation mit der absoluten Generationsnummer 0001 Basis der FGG. Während des Betriebs der Anwendung wird das Anwendungsprogramm mit KDCAPPL PROG=NEW ausgetauscht, die Generation mit der Generationsnummer 0002 wird geladen. Sie ist dann automatisch Basis der FGG. Beim nächsten Start der Anwendung lädt *open*UTM dann diese Generation (0002). Zwischen zwei Anwendungsläufen können Sie die Basis mit Hilfe der Funktion SWITCH des Tools KDCPROG umschalten:

   – Die Basis hat die relative Generationsnummmer +0000.

   – Die Generation, auf die beim Anwendungsaustausch mit KDCAPPL PROG=NEW geschaltet wird, hat die relative Generationsnummmer +0001.

- Die Generation, auf die mit KDCAPPL PROG=OLD geschaltet wird, hat die relative Generationsnummmer -0001.

2. Die Basisnummer der FGG ist 6. Dann hat die Generation mit der absoluten Generationsnummer 0001 den relativen FGG-Namen PROG(-5), die Generation mit der absoluten Generationsnummer 0008 den relativen FGG-Namen PROG(+2).
3. Die maximale Anzahl der Generationen in der FGG ist 3. In der FGG existieren im Verzeichnis PROG die Generationen 0001, 0002 und 0003:
   - 0001 ist dabei die erste Generation und 0003 die letzte Generation.
   - Beim Transfer einer neuen Generation mit der Generationsnummer 0004 in die FGG löscht *open*UTM die erste Generation 0001. Die FGG enthält damit die drei Generationen 0002, 0003 und 0004.
   - Die erste Generation ist jetzt die mit der absoluten Generationsnummer 0002. Sie wird beim Transfer einer weiteren Generation in die FGG von *open*UTM gelöscht.

8.1.3 Ablauf des Anwendungsaustausches

Den Austausch des Anwendungsprogramms veranlasst der Administrator der UTM-Anwendung z.B. mit dem Kommando KDCAPPL PROG=... . Der Programmaustausch läuft dann entkoppelt ab.

Geben Sie KDCAPPL PROG=NEW an, dann wird das Anwendungsprogramm der Generation PROG(+1) geladen. Bei KDCAPPL PROG=OLD wird das Anwendungsprogramm der Generation PROG(-1) geladen.

Aus diesem Grund ist es sinnvoll, die letzte Generation der FGG als Basis zu definieren. Diese Generation wird dann beim Start der Anwendung geladen. Eine neue Generation des Anwendungsprogramms transferieren Sie dann mit der relativen Generationsnummer PROG(+1) in die FGG (Standardeinstellung bei KDCPROG TRANSFER). Beim Programmaustausch mit KDCAPPL PROG=NEW wird dann die neue Generation des Anwendungsprogramms geladen. *open*UTM schaltet automatisch die Basis auf die aktuell geladene Generation, d.h. die letzte Generation, um. Wollen Sie dann auf das zuvor geladene Anwendungsprogramm zurückschalten, geben Sie KDCAPPL PROG=OLD an.

Der Austausch wird für jeden Workprozess der Anwendung nacheinander ausgeführt. Für jeden einzelnen Workprozess wird dazu nach Ausführung des aktuellen Auftrags das laufende Anwendungsprogramm beendet und das neue Anwendungsprogramm geladen. Erst wenn der Austausch für diesen Workprozess beendet ist, wird der Austausch für den nächsten Workprozess durchgeführt. Dadurch wird vermieden, dass der Anwendungslauf durch den Austausch nennenswert gestört wird. Der Benutzer bemerkt auf diese Weise von dem Anwendungsaustausch nichts. Er kann ungehindert weiterarbeiten.

Während des Anwendungsaustausches kann es durch den oben beschriebenen Ablauf vorkommen, dass Aufträge zur gleichen Zeit von einzelnen Workprozessen noch mit dem alten Anwendungsprogramm, von anderen Workprozessen bereits mit dem neuen Anwendungsprogramm bearbeitet werden. Das können Sie verhindern, wenn Sie vor dem Austausch die maximal zulässige Anzahl der Workprozesse auf 1 herabsetzen (z.B. mit dem Administrationskommando KDCAPPL TASKS=1).

Der UTM-Administrator wird nach Beendigung des Anwendungsaustausches durch eine Meldung informiert, ob der Austausch erfolgreich durchgeführt oder mit Fehler abgebrochen wurde. Erst wenn ein Austausch für alle Workprozesse beendet ist, kann der Administrator den nächsten Anwendungsaustausch starten.

8.1.4 Das Tool KDCPROG

Das Tool KDCPROG wird wie folgt aufgerufen:

Unter UNIX rufen Sie KDCPROG von der Shell-Ebene mit dem Kommando auf:
utm-dateiverzeichnis/ex/kdcprog *funktion operanden*

Unter Windows starten Sie die DOS-Eingabeaufforderung und geben folgendes Kommando ein:
utm-dateiverzeichnis\ex\kdcprog *funktion operanden*

KDCPROG bietet die folgenden Funktionen:

Funktion	Bedeutung
CREATE	Dateigenerationsverzeichnis PROG für den Anwendungsaustausch einrichten
INFO	Informationen über den aktuellen Zustand der FGG ausgeben
TRANSFER	Neue Version des Anwendungsprogramms in eine Generation der FGG übertragen
SWITCH	Basisnummer der FGG umschalten

Die Beschreibung der Operanden finden Sie auf den folgenden Seiten.

CREATE - Dateigenerationsverzeichnis (FGG) einrichten

KDCPROG CREATE erzeugt eine FGG für den Anwendungsaustausch. Es wird ein Dateiverzeichnis PROG im Basisverzeichnis *filebase* der Anwendung angelegt. Ein bereits bestehendes Verzeichnis PROG wird von KDCPROG vorher vollständig gelöscht.

KDCPROG␣CREATE␣*filebase*␣*number_entries*

filebase Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

number_entries
Maximale Anzahl Generationen, die gleichzeitig in der FGG PROG existieren können. Sobald *number_entries* in der FGG vorhanden sind, wird beim Transfer einer neuen Generation in die FGG die erste Generation der FGG gelöscht.

Minimalwert: 2
Maximalwert: 9999

INFO - Aktuellen Zustand der FGG abfragen

KDCPROG INFO zeigt den aktuellen Zustand der FGG an, wobei die folgenden Daten ausgegeben werden:

– aktuelle Anzahl der Einträge

– Basisnummer der FGG

– Dateigeneration mit der niedrigsten Generationsnummer (erste Generation)

– Dateigeneration mit der höchsten Generationsnummer (letzte Generation)

– Liste der in der FGG enthaltenen Dateigenerationen mit absoluten und relativen Namen

– Liste der vorhandenen Generationen im Verzeichnis PROG. Die Ausgabe entspricht in UNIX der Ausgabe des Kommandos `ls -l` für das Dateiverzeichnis PROG.
In Windows wird ein allgemeinerer Hinweis auf das Kommando `dir` ausgegeben.

Beispiele für die Ausgabe von KDCPROG INFO finden Sie in Abschnitt „Beispiel für Anwendungsaustausch“ auf Seite 132.

`KDCPROG␣INFO␣`*filebase*

filebase Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

TRANSFER - utmwork in die FGG übertragen

KDCPROG TRANSFER überträgt *utmwork* und die zugehörige *nmutmwork*-Datei aus *filebase* in die FGG.

KDCPROG␣TRANSFER␣*filebase*␣*generationnumber*

filebase
: Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

generationnumber
: Nummer der Generation, in welche *utmwork* und *nmutmwork* übertragen werden sollen.

  Die Angabe von *generationnumber* ist beim Transfer der ersten Version des Anwendungsprogramms Pflicht. Für *generationnumber* müssen Sie 0001 (absolut) oder +0 (relativ) angeben. Bei folgenden Transfers ist die Angabe optional. Geben Sie *generationnumber* nicht an, nimmt *open*UTM den Wert +1 an.

  Wird für *generationnumber* eine Generation angegeben, die bereits in der FGG vorhanden ist, so wird diese Generation überschrieben.

  Die Angabe von *generationnumber* ist auf zwei Arten möglich:

  1. Angabe einer absoluten Generationsnummer.

     Maximal- und Minimalwert von *generationnumber* hängen von der Anzahl der FGG-Einträge ab.

     Die erste Generation, die Sie in die FGG transferieren, muss immer die absolute Generationsnummer 0001 haben. Neue Generationsnummern, die Sie danach vergeben, müssen fortlaufend ansteigend sein (0002, 0003...). Sie können auch Generationsnummern von Generationen angeben, die bereits in der FGG vorhanden sind. Diese werden dann überschrieben.

     KDCPROG gibt folgende absolute Grenzwerte vor:

     Minimalwert: 1
     Maximalwert: 9999

Hinweis

Die Angabe (aktuelle Basisnummer - 1) für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=OLD geschaltet wird.
Die Angabe (aktuelle Basisnummer + 1) für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=NEW geschaltet wird.

2. Angabe einer relativen Generationsnummer mit führendem Vorzeichen (+ oder -). Maximal- und Minimalwert von *generationnumber* hängen vom aktuellen Basiswert und der Anzahl der FGG-Einträge ab. Ist beispielsweise die letzte Generation (höchste absolute Generationsnummer) Basis, dann dürfen Sie keine relative Generationsnummer angeben, die größer als +1 ist.

   KDCPROG gibt folgende absolute Grenzwerte vor:

   Minimalwert: - 99
   Maximalwert: + 99

   Hinweis

   Die Angabe -1 für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=OLD geschaltet wird.
   Die Angabe +1 für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=NEW geschaltet wird.

   Standardwert: +1 (relative Generationsnummer)

   Mit dem Standardwert wird die neue Version des Anwendungsprogramms in den FGG-Eintrag transferiert, auf die mit KDCAPPL PROG=NEW geschaltet wird.

SWITCH - Basis der FGG umschalten

Mit KDCPROG SWITCH können Sie außerhalb des Anwendungslaufs die Basis der FGG umschalten. Beim nächsten Start der Anwendung wird dann die neue Basis der FGG geladen. D.h. mit KDCPROG SWITCH können Sie zwischen zwei Anwendungsläufen die Funktionen von KDCAPPL PROG=NEW bzw. PROG=OLD ausführen.

Der Aufruf von KDCPROG SWITCH während des Anwendungslaufs wird abgelehnt.

KDCPROG␣SWITCH␣*filebase*␣*basenumber*

filebase Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

basenumber Angabe der neuen Basisgeneration. Für *basenumber* dürfen Sie nur eine Generationsnummer angeben, zu der schon eine Generation in der FGG existiert.

Die Angabe kann auf zwei Arten erfolgen:

1. Angabe einer absoluten Generationsnummer

   basenumber bezeichnet die neue Basisgeneration direkt.

   Minimalwert: 0
   Maximalwert: 9999

2. Angabe einer relativen Generationsnummer mit negativem Vorzeichen
   Die Basisgeneration wird relativ zur letzten Generation (= Generation mit der höchsten Generationsnummer) angegeben.
   Der Wert von *basenumber* muss in diesem Fall immer mit führendem Minuszeichen (-) angegeben werden.

   Generationsnummer der Basis =
   Nummer der letzten Generation - *basenumber*

   Minimalwert: -99
   Maximalwert: -1

Beispiel

Die Generation mit der Geneationsnummer 0010 ist die letzte Generation in der FGG.
KDCPROG SWITCH *filebase* -1 bewirkt, dass die Dateigeneration mit der Generationsnummer 0009 neue Basis der FGG ist.
KDCPROG SWITCH *filebase* 0 bewirkt, dass die letzte Generation (0010) neue Basis der FGG ist.

8.1.5 Beispiel für Anwendungsaustausch

In den folgenden Abschnitten wird ein Beispiel-Anwendungsaustausch mit dem Tool KDCPROG durchgeführt.

1. Arbeitsschritt

Es wird eine FGG zum Anwendungsaustausch erzeugt, die maximal drei Generationen des Anwendungsprogramms enthalten darf. Danach wird die erste Generation des Anwendungsprogramms in die FGG transferiert. Diese Generation ist dann die Basis der FGG. Für *filebase* wird das aktuelle Dateiverzeichnis („.“) angegeben.

In UNIX:

Eingabe:
```
utm-dateiverzeichnis/ex/kdcprog CREATE . 3
```

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U376 kdcprog: FGG-Dateien fuer ./PROG erzeugt.
```

Eingabe:
```
utm-dateiverzeichnis/ex/kdcprog TRANSFER . +1
utm-dateiverzeichnis/ex/kdcprog SWITCH . 1
```

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0001
U389 kdcprog: TRANSFER erfolgreich
U388 kdcprog: neue Basis der Programm FGG./PROG ist 1
```

In Windows:

Eingabe:
```
utm-dateiverzeichnis\ex\kdcprog CREATE . 3
```

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U376 kdcprog: FGG-Dateien fuer ./PROG erzeugt (pid: 348,...)
```

Eingabe:
```
utm-dateiverzeichnis\ex\kdcprog TRANSFER .
```

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet (pid: 420,...)
U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen
U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0001
 1 Datei(en) kopiert.
U389 kdcprog: TRANSFER erfolgreich
```

Eingabe:
```
utm-dateiverzeichnis\ex\kdcprog SWITCH . 1
```

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet (pid: 192,...)
U388 kdcprog: neue Basis der Programm FGG./PROG ist 1
```

2. Arbeitsschritt

Während des Anwendungslaufs wird eine neue Version des Anwendungsprogramms erstellt. Diese Version soll als nächste Generation (Generationsnummer 0002) in die FGG transferiert werden. Die relative Generationsnummer dieser Generation ist dann (+1). Dies ist die Standardeinstellung beim TRANSFER.

In UNIX:

Eingabe: *utm-dateiverzeichnis*/ex/kdcprog TRANSFER .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0002
U389 kdcprog: TRANSFER erfolgreich
```

In Windows:

Eingabe: *utm-dateiverzeichnis*\ex\kdcprog TRANSFER .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet (pid: 234,...)
U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen
U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0002
 1 Datei(en) kopiert.
U389 kdcprog: TRANSFER erfolgreich
```

Die transferierte Generation wird bei KDCAPPL PROG=NEW benutzt.

3. Arbeitsschritt

Es wird Information über die FGG angefordert.

In UNIX:

Eingabe: *utm-dateiverzeichnis*/ex/kdcprog INFO .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen          3
         FGG Basis                  0001
         FGG erste Generation       0001
         FGG letzte Generation      0002
Datei PROG/0001 ist PROG(+0000) <=
Datei PROG/0002 ist PROG(+0001)
Die folgenden Programmdateien sind verfuegbar:
-rwx------  1 beispiel  other    2845876 Apr 22 15:35 ./PROG/0001
-rwx------  1 beispiel  other    2845876 Apr 22 15:37 ./PROG/0002
```

In Windows:

Eingabe: *utm-dateiverzeichnis*\ex\kdcprog INFO .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet (pid 480 ...)
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen           3
         FGG Basis                   0001
         FGG erste Generation        0001
         FGG letzte Generation       0002
Datei PROG/0001 ist PROG(+0000) <=
Datei PROG/0002 ist PROG(+0001)
Die folgenden Programmdateien sind verfuegbar:
kdcprog: type "DIR .\PROG\*.EXE" to get full information
```

Bei der Ausgabe ist zu beachten, dass PROG/000*x* den Namen der jeweiligen Generation angibt. PROG(+000*x*) ist der relative FGG-Name. Der Pfeil „<=“ zeigt auf die Basis der FGG; das ist die aktuell geladene Generation des Anwendungsprogramms.

4. Arbeitsschritt

Der UTM-Administrator führt einen Anwendungsaustausch durch. Es soll die Generation 0002 (alias PROG(+1)) geladen werden. Dazu meldet sich der Administrator bei der UTM-Anwendung an und gibt z.B. ein:

```
KDCAPPL PROG=NEW
```

Nach dem Austausch wird noch einmal Information über die FGG angefordert.

In UNIX:

Eingabe: *utm-dateiverzeichnis*/ex/kdcprog INFO .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen           3
         FGG Basis                   0002
         FGG erste Generation        0001
         FGG letzte Generation       0002
Datei PROG/0001 ist PROG(-0001)
Datei PROG/0002 ist PROG(+0000) <=
Die folgenden Programmdateien sind verfuegbar:
-rwx------   1 beispiel  other    2845876 Apr 22 15:35 ./PROG/0001
-rwx------   1 beispiel  other    2845876 Apr 22 15:37 ./PROG/0002
```

In Windows:

Eingabe: *utm-dateiverzeichnis*\ex\kdcprog INFO .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen          3
         FGG Basis                  0002
         FGG erste Generation       0001
         FGG letzte Generation      0002
Datei PROG/0001 ist PROG(-0001)
Datei PROG/0002 ist PROG(+0000) <=
Die folgenden Programmdateien sind verfuegbar:
kdcprog: type "DIR .\PROG\*.EXE" to get full information
```

Die Ausgabe zeigt, dass *open*UTM die Basis geändert hat. Basis ist nun die Generation mit der Generationsnummer 0002, die beim Anwendungsaustausch geladen wurde.
Die Generation 0001 wird benutzt, falls KDCAPPL PROG=OLD eingegeben wird. Für KDCAPPL PROG=NEW ist kein Programm vorhanden.

5. Arbeitsschritt

Es wird eine weitere Version des Anwendungsprogramms in die FGG transferiert. Damit steht für einen erneuten Anwendungsaustausch mit KDCAPPL PROG=NEW eine Version des Anwendungsprogramms zur Verfügung.

Nach dem Transfer wird erneut Information über die FGG angefordert.

In UNIX:

Eingabe: *utm-dateiverzeichnis*/ex/kdcprog TRANSFER .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0003
U389 kdcprog: TRANSFER erfolgreich
```

Eingabe *utm-dateiverzeichnis*/ex/kdcprog INFO .

Ausgabe
```
U181 Programm kdcprog V05.2A00 gestartet
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen          3
         FGG Basis                   0002
         FGG erste Generation        0001
         FGG letzte Generation       0003
Datei PROG/0001 ist PROG(-0001)
Datei PROG/0002 ist PROG(+0000) <=
Datei PROG/0003 ist PROG(+0001)
Die folgenden Programmdateien sind verfuegbar:
-rwx------   1 beispiel other    2845876 Apr 22 15:35 ./PROG/0001
-rwx------   1 beispiel other    2845876 Apr 22 15:37 ./PROG/0002
-rwx------   1 beispiel other    2845876 Apr 22 15:43 ./PROG/0003
```

In Windows:

Eingabe: *utm-dateiverzeichnis*\ex\kdcprog TRANSFER .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet (pid: 261,...)
U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen
U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0003
 1 Datei(en) kopiert.
U389 kdcprog: TRANSFER erfolgreich
```

Eingabe *utm-dateiverzeichnis*\ex\kdcprog INFO .

Ausgabe
```
U181 Programm kdcprog V05.2A00 gestartet
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen          3
         FGG Basis                   0002
         FGG erste Generation        0001
         FGG letzte Generation       0003
Datei PROG/0001 ist PROG(-0001)
Datei PROG/0002 ist PROG(+0000) <=
Datei PROG/0003 ist PROG(+0001)
Die folgenden Programmdateien sind verfuegbar:
kdcprog: type "DIR .\PROG\*.EXE" to get full information
```

Jetzt steht je ein Programm für einen Anwendungsaustausch mit KDCAPPL PROG=OLD und mit KDCAPPL PROG=NEW zur Verfügung.

6. Arbeitsschritt

Es wird eine weitere Version des Anwendungsprogramms in die FGG transferiert und KDCPROG INFO aufgerufen.

In UNIX:

Eingabe:
```
utm-dateiverzeichnis/ex/kdcprog TRANSFER .
```

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet
U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0004
U389 kdcprog: TRANSFER erfolgreich
```

Eingabe
```
utm-dateiverzeichnis/ex/kdcprog INFO .
```

Ausgabe
```
U181 Programm kdcprog V05.2A00 gestartet
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen          3
         FGG Basis                   0002
         FGG erste Generation        0002
         FGG letzte Generation       0004
Datei PROG/0002 ist PROG(+0000) <=
Datei PROG/0003 ist PROG(+0001)
Datei PROG/0004 ist PROG(+0002)
Die folgenden Programmdateien sind verfuegbar:
-rwx------   1 beispiel   other    2845876 Apr 22 15:37 ./PROG/0002
-rwx------   1 beispiel   other    2845876 Apr 22 15:43 ./PROG/0003
-rwx------   1 beispiel   other    2845876 Apr 22 15:59 ./PROG/0004
```

In Windows:

Eingabe: *utm-dateiverzeichnis*\ex\kdcprog TRANSFER .

Ausgabe:
```
U181 Programm kdcprog V05.2A00 gestartet (pid: 261,...)
U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen
U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0004
 1 Datei(en) kopiert.
U389 kdcprog: TRANSFER erfolgreich
```

Eingabe *utm-dateiverzeichnis*\ex\kdcprog INFO .

Ausgabe
```
U181 Programm kdcprog V05.2A00 gestartet
U378    INFO fuer FGG ./PROG
         FGG Maximal Anzahl Versionen          3
         FGG Basis                   0002
         FGG erste Generation        0002
         FGG letzte Generation       0004
Datei PROG/0002 ist PROG(+0000) <=
Datei PROG/0003 ist PROG(+0001)
Datei PROG/0004 ist PROG(+0002)
Die folgenden Programmdateien sind verfuegbar:
kdcprog: type "DIR .\PROG\*.EXE" to get full information
```

Die Generation 0001 wurde gelöscht, da maximal drei Generationen in der FGG enthalten sein dürfen.

8.2 Shared Objects austauschen

Mit der Funktion „Shared Objects austauschen“ können Sie während des laufenden Betriebs einzelne Teile des Anwendungsprogramms austauschen. Diese Anwendungsteile müssen als Shared Objects erzeugt und dynamisch zur Anwendung dazugebunden werden. Dabei müssen Sie bestimmte Schritte beim Übersetzen, Binden und Generieren vornehmen.

In Windows werden Shared Objects mit Hilfe von DLLs realisiert. Einzelheiten dazu finden Sie im Abschnitt „Anwendungsprogramme als DLLs erstellen“ auf Seite 30.

Shared Objects können Sie mit dem Administrationskommando KDCPROG austauschen oder mit einem eigenen Administrationsprogramm, das KDCADMI mit Operationscode KC_MODIFY_OBJECT und Objekttyp KC_LOAD_MODULE aufruft.
Beide Möglichkeiten sind im *open*UTM-Handbuch „Anwendungen administrieren“ beschrieben.

Wenn im Folgenden auf Aktionen hingewiesen wird, die mit dem Kommando KDCPROG durchgeführt werden können, dann gilt dies auch für Administrationsprogramme, die KDCADMI mit Operationscode KC_MODIFY_OBJECT und Objekttyp KC_LOAD_MODULE absetzen.

Eine Anwendung mit Shared Objects kann auch als Ganzes ausgetauscht werden.

8.2.1 Shared Objects bereitstellen und generieren

Ein Shared Object in C müssen Sie in UNIX immer so übersetzen, dass dabei das jeweilige Laufzeitsystem mit eingebunden wird. Näheres über das Übersetzen in Windows siehe Seite 26.

Versionskonzept von Shared Objects

Shared Objects können mit oder ohne Versionen erstellt werden.

- Ohne Versionen

  Wenn Sie ein Shared Object ohne Version bereitstellen wollen, dann müssen Sie genau eine Datei mit dem Shared Object bereitstellen. Beim Austauschen mit dem Administrationskommando KDCPROG reicht die Angabe des Dateinamens. Shared Objects ohne Version können nur beim Start der Anwendung nachgeladen werden.

- Mit Versionen

  Soll ein Shared Object in mehreren Versionen verfügbar sein, dann müssen Sie zuerst ein Verzeichnis anlegen und anschließend die einzelnen Versionen des Shared Object in dieses Verzeichnis kopieren. Sie können beliebig viele Versionen einbringen. Beim Austauschen geben Sie sowohl den Verzeichnisnamen des Shared Objects als auch den Versionsnamen an.

  In Windows sollten Sie Shared Objects immer **mit** Versionen erstellen.

Shared Objects generieren

Jedes Shared Object muss mit der KDCDEF-Anweisung SHARED-OBJECT generiert werden (siehe *open*UTM-Handbuch „Anwendungen generieren“) . Dabei geben Sie Folgendes an:

- Den Namen, den das Shared Object besitzt. Ist es ein Shared Object ohne Versionen, dann geben Sie den Dateinamen an, unter dem es abgespeichert ist. Ist es ein Shared Object mit Versionen, dann geben Sie den Namen des Verzeichnisses an, in dem die Versionen stehen.
  Pro Shared Object kann nur eine Version generiert werden. Die Version kann per UTM-Administration geändert werden.
- Den Dateinamen der jeweiligen Version, falls es ein Shared Object mit Version ist (Operand VERSION).
- Den Pfadnamen, unter dem das Shared Object zu finden ist (Operand DIRECTORY).

  In Windows sollten Sie immer den kompletten Pfad angeben, da die Umgebungsvariablen PATH und LD_LIBRARY_PATH für Shared Objects nicht ausgewertet werden.
- Ob das Shared Object beim Start der Anwendung (LOAD-MODE=STARTUP) oder beim ersten Aufruf (LOAD-MODE=ONCALL) geladen werden soll.

In der zu dem Teilprogramm gehörenden PROGRAM-Anweisung müssen Sie den Namen des Shared Object angeben (Operand SHARED-OBJECT, siehe auch die Beispiele auf Seite 143).

8.2.2 Start der Anwendung

Beim Start der Anwendung lädt *open*UTM alle Shared Objects, die mit LOAD=STARTUP generiert sind, und zwar in der Reihenfolge, in der die SHARED-OBJECT-Anweisungen gegeben wurden. Shared Objects mit LOAD-MODE=ONCALL werden erst beim erstmaligen Aufruf geladen.

Kann ein Shared Object nicht geladen werden, dann wird der Start dennoch fortgesetzt. Wird ein solches Shared Object später aufgerufen, dann führt dies zu einem BADTAC oder einem PEND ER.

Können die Event-Exits START, SHUT oder INPUT bzw. die Event-Services MSGTAC oder SIGNON oder das Administrationsteilprogramm KDCADM nicht geladen werden, dann wird der Start mit einer Fehlermeldung abgebrochen.

8.2.3 Ablauf des Austausches

Den Austausch eines Shared Objects muss der *open*UTM-Administrator veranlassen, z.B. mit dem Administrationskommando KDCPROG. Dabei werden die Event-Exits START und SHUT nicht durchlaufen, es sei denn, durch den Austausch wird das Anwendungsprogramm beendet und neu geladen.

Wenn Sie ein Shared Object austauschen, das mit Versionen generiert wurde, dann müssen Sie den Verzeichnisnamen und den Versionsnamen angeben. Bei Shared Objects ohne Versionen geben Sie den Namen des Shared Objects selber an, eine Versionsangabe wird ignoriert.

Der Austausch verläuft unterschiedlich, je nachdem, zu welchem Zeitpunkt (STARTUP oder ONCALL) das Shared Object geladen wurde.

8.2.3.1 Shared Objects mit LOAD-MODE=STARTUP austauschen

Beim Austausch von Anwendungsteilen, die mit LOAD-MODE=STARTUP generiert wurden, läuft der Workprozess weiter. Das Shared Object wird entladen und die angegebene Version neu geladen.

Diesen Programmaustausch können mehrere Workprozesse einer Anwendung gleichzeitig ausführen. Während des Programmaustauschs sind in den Workprozessen der UTM-Anwendung unterschiedliche Stände des Anwendungsprogramms geladen. Jeder Workprozess der Anwendung führt den angeforderten Programmaustausch am Ende der Bearbeitung des aktuellen Auftrags durch. Das Ende des Programmaustauschs wird durch eine Meldung angezeigt.

Bis der Programmaustausch abgeschlossen ist, kann kein weiterer Programmaustausch gestartet werden. Ein erneuter Aufruf von KDCPROG wird von *open*UTM abgewiesen.

8.2.3.2 Shared Objects mit LOAD-MODE=ONCALL austauschen

Shared Objects, die mit LOAD-MODE=ONCALL generiert sind, können Sie nur austauschen, wenn diese **mit** Versionen erzeugt wurden.

Wenn Sie ein solches Shared Object austauschen, dann wird bei der Bearbeitung des Administrationskommandos KDCPROG nur der neu zu ladende Versionsbezeichner des betroffenen Shared Objects in die *open*UTM-Tabellen eingetragen.

Die neue Version wird von jedem Workprozess der Anwendung erst dann geladen, wenn dieser Workprozess das nächste Mal ein Teilprogramm aufruft, das in diesem Shared Object enthalten ist. Diesen Programmaustausch können mehrere Workprozesse einer Anwendung gleichzeitig ausführen. Bis der angeforderte Programmaustausch von allen Workprozessen der UTM-Anwendung durchgeführt wurde, sind in den einzelnen Workprozessen unterschiedliche Stände des Anwendungsprogramms geladen. Es ist jedoch sichergestellt, dass jeder Workprozess den angeforderten Austausch durchführt, bevor erneut ein Teilprogramm aktiviert wird, das in dem auszutauschenden Shared Object enthalten ist.

Der Austausch eines mit ONCALL generierten Shared Objects wirkt nicht blockierend auf nachfolgende Kommandos zum Programmaustausch. D.h. der Administrator kann unmittelbar nach der Bearbeitung des Kommandos KDCPROG mit einem weiteren Kommando KDCPROG einen neuen Programmaustausch veranlassen.

Stimmen die Versionsbezeichner von neuem und altem Shared Object überein, wird kein Programmaustausch durchgeführt.

8.2.4 Beispiele für Austausch von Shared Objects

Beispiel 1

Auf einem UNIX-System soll ein einzelnes Modul mit Namen EKSTEUER dynamisch zu einem bestimmten Stichtag ausgetauscht werden. Dazu gehen Sie wie folgt vor:

1. Erstellen und Übersetzen Sie das Modul mit den Optionen, die für Shared Objects notwendig sind.
2. Bringen Sie es als Shared Object ohne Version unter dem Namen EKSTEUER in das Verzeichnis, in dem sich die benutzereigenen Programme befinden. Im Beispiel wird für das Verzeichnis der Platzhalter *so-lib* angegeben, das kann z.B. ein Verzeichnis `/usr/proglib` (in UNIX) oder `C:\proglib` (in Windows) sein.
3. Generieren Sie das Modul mit folgenden KDCDEF-Anweisungen als Shared Object:

   ```
   SHARED-OBJECT EKSTEUER,DIRECTORY=so-lib,LOAD-MODE=STARTUP
   PROGRAM .... ,SHARED-OBJECT=EKSTEUER
   ```

   Das Shared Object wird damit beim Anwendungsstart geladen (Pflicht bei Shared Objects ohne Version).
4. Binden Sie den Workprozess, wobei Sie die dynamische Bibliothek mit dem Shared Object EKSTEUER angeben müssen.
5. Starten Sie die Anwendung wie gewohnt.
6. Ändern Sie das Modul und speichern Sie es vor dem Stichtag in die Datei:

   so-lib/EKSTEUER
7. Geben Sie folgendes Administrationskommando ein:

   ```
   KDCPROG SHARED-OBJECT=EKSTEUER
   ```

   Das Shared Object wird in den einzelnen Workprozessen ausgetauscht, sobald diese den aktuellen Auftrag abgearbeitet haben.

Beispiel 2

Auf einem UNIX- bzw. Windows-System soll ein Shared Object mit Namen MONATSABSCHLUSS in 12 Versionen (BIL01,... BIL12) vorhanden sein und zu jedem Monatswechsel ausgetauscht werden. Es soll immer erst beim erstmaligen Aufruf geladen werden. Dazu gehen Sie wie folgt vor:

1. Erstellen und Übersetzen Sie das Modul mit den Optionen, die für Shared Objects notwendig sind (für Windows siehe Seite 30).
2. Richten Sie im Verzeichnis mit den benutzereigenen Programmen (im Beispiel *so-lib*) das Verzeichnis MONATSABSCHLUSS ein und kopieren Sie zumindest die beim ersten Anwendungslauf benötigte Version in dieses Verzeichnis.
3. Generieren Sie jede Version des Shared Object mit folgender KDCDEF-Anweisung:

   ```
   SHARED-OBJECT MONATSABSCHLUSS
                 ,DIRECTORY=so-lib
                 ,VERSION=BILxx       (xx=01,... ,12)
                 ,LOAD-MODE=ONCALL
   ```

   Das Shared Object wird damit erst beim Aufruf des Teilprogramms geladen.

   Geben Sie für das Shared Object folgende PROGRAM-Anweisung:

   ```
   PROGRAM .... ,SHARED-OBJECT=MONATSABSCHLUSS
   ```

4. In UNIX binden Sie den Workprozess, indem Sie die dynamische Bibliothek mit den Shared Objects angeben.
   In Windows gibt es an dieser Stelle keine Besonderheiten.
5. Starten Sie die Anwendung wie gewohnt.
6. Geben Sie z.B. am 1. Juli folgendes Administrationskommando ein:

   ```
   KDCPROG SHARED-OBJECT=MONATSABSCHLUSS,VERSION=BIL07
   ```

   Achten Sie bitte darauf, dass diese Version zum genannten Zeitpunkt auch im Verzeichnis vorhanden ist.

   Das Shared Object wird in den einzelnen Workprozessen erst ausgetauscht, wenn das entsprechende Teilprogramm erstmals aufgerufen wird.

8.2.5 Anwendung mit Shared Objects austauschen

Die gesamte Anwendung können Sie mit dem Administrationskommando KDCAPPL PROG=NEW austauschen. Der Austausch kann entweder über das Dateigenerationsverzeichnis PROG erfolgen, das Sie mit dem Tool KDCPROG vorbereiten müssen (siehe Abschnitt „Voraussetzungen für den Anwendungsaustausch“ auf Seite 120) oder UTM lädt das Anwendungsprogramm *utmwork* direkt aus dem Basisverzeichnis.

In beiden Fällen wird beim Austausch der gesamten Anwendung nacheinander jeder Workprozess der Anwendung entladen und anschließend neu geladen. Beim Neuladen werden die neuen Versionen der Shared Objects geladen. Um den Betrieb der Anwendung möglichst wenig zu stören, tauscht zu einer Zeit immer nur ein Workprozess der Anwendung.

8.2.6 Programme dynamisch hinzufügen

Per dynamischer Administration können u.a. Programme im Betrieb der Anwendung neu generiert werden. Näheres zur dynamischen Administration siehe *open*UTM-Handbuch „Anwendungen administrieren“.

Bevor diese Programme aufgerufen werden können, müssen sie zunächst geladen werden. Dazu muss das Programm zu dem zugeordneten Shared Object gebunden und mit einer neuen Version im Verzeichnis bereitgestellt werden, das in der SHARED-OBJECT-Anweisung beim Generieren angegeben wurde.

Anschließend muss der Administrator dieses Shared Object durch das Kommando KDCPROG oder durch Programmaufruf austauschen.

Der UTM-Administrator muss die neuen Teilprogramme und die zugehörigen Transaktionscodes dynamisch in die KDCFILE-Tabellen eintragen.

9 Fehlertoleranz von *open*UTM

Fehlertoleranz heißt hier, dass eine UTM-Anwendung auch dann betriebsbereit bleibt, wenn in einzelnen Teilprogrammen Fehler auftreten, die *open*UTM zum Abbruch einer Transaktion zwingen. *open*UTM sorgt dann dafür, dass das Anwendungsprogramm beendet und neu geladen wird, so dass sich der Fehler nicht weiter ausbreitet und andere Benutzer der Anwendung und deren Daten dadurch nicht beeinträchtigt werden.

Beim Fehlerverhalten von *open*UTM unterscheidet man:

- Interne UTM-Fehler und Fehler in der Systemumgebung

  Sie führen zum abnormalen Beenden der Anwendung, ebenso wie das Administra-tionskommando KDCSHUT KILL oder das Absetzen eines KDCADMI-Aufrufs mit Operationscode KC_SHUTDOWN und Subcode KC_KILL.
  *open*UTM erzeugt für jeden Prozess der Anwendung einen UTM-Dump. Der UTM-Dump wird mit dem Tool KDCDUMP aufbereitet. Wie das geht, ist im *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“ beschrieben.

  – Bei schweren Fehlern im Dialogterminalprozess beendet sich dieser und schreibt unter dem aktuellen Dateiverzeichnis einen Core-Dump. Ein erneutes Anmelden während dieses Anmeldungslaufes ist vom zugeordneten Terminal aus nicht mehr möglich. Bei leichten Fehlern meldet sich der Dialogterminalprozess ordnungsgemäß von der Anwendung ab.

  – Ein Druckerprozess verhält sich bei Fehlern ähnlich wie ein Dialogterminalprozess. Er kann ggf. über Administrationskommando neu gestartet werden.

  – Bei Fehlern im Timerprozess wird die Anwendung abnormal beendet, sobald von den Workprozessen ein Auftrag an den Timerprozess ergeht.

- Fehler im Anwendungsprogramm

  Dabei handelt es sich um Fehler in Teilprogrammen, die sich in zwei Gruppen einteilen lassen:

  – Fehler, die zum Neuladen des Anwendungsprogramms führen

  – Fehler, die gegebenenfalls eine Fortsetzung des Programms erlauben

9.1 Fehler, die *open*UTM erkennt

Ein Teilprogramm wird in folgenden Fällen durch *open*UTM abnormal beendet:

- Es wurde ein PEND ER oder FR programmiert.
- Ein UTM-Aufruf hat einen KDCS-Returncode ≥ 70Z geliefert. In diesem Fall setzt *open*UTM intern PEND ER.

In allen Fällen bricht *open*UTM also den Vorgang ab. Falls PEND FR programmiert war, veranlasst *open*UTM keine weitere Aktionen.

Falls der Vorgang durch PEND ER (per Programm oder intern) beendet wurde, erzeugt *open*UTM einen UTM-Dump mit REASON=PENDER, der nur die Daten des KDCROOT wiedergibt. Anschließend wird der betroffene Workprozess beendet. Der Mainprozess startet danach einen neuen Workprozess, der das Anwendungsprogramm neu lädt. Dadurch wird auf einem neuen Stand der statischen Datenbereiche aufgesetzt und es werden Folgefehler durch überschriebene Daten vermieden.

9.2 Reaktion von *open*UTM auf Signale

Beim Auftreten eines Signals sind die folgenden Reaktionen möglich:

- Ignorieren dieses Signals (siehe Tabelle).
- Wenn das Signal während des Ablaufs von Anwender-Programmteilen auftritt:
  Es wird ein PEND ER mit KCRCCC=70Z und KCRCDC=XT*xx* aufgerufen
  (*xx* ist die Signalnummer).
  Der betreffende Vorgang bzw. Workprozess wird beendet.
- Wenn das Signal während des Ablaufs von UTM-Programmteilen auftritt:
  Es führt zum abnormalen Anwendungsende mit REASON=SIG*xxx*
  (*xxx* ist die Signalnummer).

Die folgende Tabelle zeigt die Reaktion eines Workprozesses auf die einzelnen Signale. Detaillierte Informationen zu den Signalen sind dem C-Headerfile für Signale (`signal.h`) zu entnehmen.

Empfangenes Signal	**Reaktion**
SIGHUP	wird ignoriert
SIGINT	wird ignoriert
SIGQUIT	wird ignoriert
SIGILL	PEND ER/TRMA[1]

Empfangenes Signal	Reaktion
SIGTRAP	wird ignoriert
SIGABRT	wird ignoriert
SIGEMT	wird ignoriert
SIGFPE	PEND ER/TRMA[1]
SIGBUS	PEND ER/TRMA[1]
SIGSEGV	PEND ER/TRMA[1]
SIGSYS	PEND ER/TRMA[1]
SIGPIPE	wird ignoriert
SIGALRM	wird ignoriert
SIGTERM	wird ignoriert
SIGUSR1	wird ignoriert
SIGUSR2	wird ignoriert
SIGCHLD	wird ignoriert
SIGPWR	PEND ER/TRMA[1]
SIGWINCH	wird ignoriert
SIGURG	wird ignoriert
SIGIO	wird ignoriert
SIGTSTP	wird ignoriert
SIGCONT	wird ignoriert
SIGTTIN	wird ignoriert
SIGTTOU	wird ignoriert
SIGVTALRM	wird ignoriert
SIGPROF	wird ignoriert
SIGXCPU	PEND ER/TRMA[1]
SIGXFSZ	PEND ER/TRMA[1]

[1] TRMA steht für Term Application (= Anwendungsabbruch)

Tritt während der durch ein Signal hervorgerufenen Endebehandlung eine Unterbrechung durch ein weiteres Signal auf, dann wird dieses Signal nicht von *open*UTM abgefangen. In diesem Fall übernimmt das Betriebssystem die Behandlung der Signalunterbrechung.

9.3 Anwendungsende durch System-Absturz/Shutdown

Ein System-Absturz bzw. ein Betriebssystem-Shutdown führt zu einem abnormalen Anwendungsende, bei dem jedoch kein UTM-Dump erzeugt wird. Sämtliche Prozesse der Anwendung werden vom Betriebssystem beendet. Bevor die Anwendung in solchen Fällen wieder gestartet wird, muss das Tool KDCREM aufgerufen werden. Sehen Sie dazu den Abschnitt „Das Tool KDCREM“ auf Seite 72.

10 Accounting

*open*UTM stellt Accounting-Funktionen zur Verfügung, die es dem Betreiber einer UTM-Anwendung ermöglichen, den Benutzern der UTM-Anwendung die in Anspruch genommenen Betriebsmittel zu verrechnen. Die Accounting-Funktionen, die das jeweilige Betriebssystem zur Verfügung stellt, können die Betriebsmittelauslastung und Leistung einer UTM-Anwendung nur als Ganzes erfassen. Wenn Sie die DV-Leistungen aber den einzelnen Benutzern der UTM-Anwendung zuordnen und in Rechnung stellen wollen, muss für das UTM-Accounting Folgendes berücksichtigt werden:

- Die Benutzer einer UTM-Anwendung werden durch Benutzerkennungen der UTM-Generierung repräsentiert und nicht durch Benutzerkennungen des Betriebssystems. Die von einem Benutzer in Anspruch genommenen Leistungen müssen also den einzelnen UTM-Benutzerkennungen zugeordnet werden können.
- In einer UTM-Anwendung ist eine Gruppe homogener Prozesse aktiv. Jeder Prozess bearbeitet nacheinander Aufträge für wechselnde Benutzer. Die innerhalb eines Prozesses in Anspruch genommenen Leistungen müssen deshalb pro aufgerufenen Service (d.h. für einzelne Teilprogrammläufe) ermittelt werden.
- Die Zeitbedingungen des OLTP-Betriebs erfordern eine Form der Leistungserfassung, die die Performance der Anwendung nicht beeinträchtigt.

Beim UTM-Accounting wird also der Verbrauch an Betriebsmitteln erfasst, der von den einzelnen Teilprogrammen beansprucht wird. Damit kann der Betriebsmittelverbrauch dem Transaktionscode (TAC) des jeweiligen Teilprogramms und somit auch dem UTM-Benutzer, der den zugehörigen Service gestartet hat, zugeordnet werden.

Über die vom UTM-Accounting erfassten Leistungen hinaus gibt es einen Grundbedarf an Betriebsmitteln, der beim Ablauf einer UTM-Anwendung anfällt, aber nicht direkt einem Benutzer zugeordnet werden kann. Das sind:

- Plattenbelegung für KDCFILE-, SYSLOG- und USLOG-Dateien sowie
- CPU-Verbrauch für
  - Starten und Beenden der UTM-Prozesse
  - Verbindungsbehandlung der zugeordneten Clients
  - LPUT-Behandlung (Übertragung auf USLOG-Datei)
  - Aufbereitung von Druckerausgaben

Sollen diese Leistungen bei der Abrechnung berücksichtigt werden, dann müssen Sie sie den Benutzern als Pauschale in Rechnung stellen.

10.1 Begriffsdefinitionen

In diesem Abschnitt werden einige Begriffe genauer erläutert, die für das UTM-Accounting relevant sind.

Benutzer im Sinne des UTM-Accounting

Der Benutzer einer UTM-Anwendung, für den eine Abrechnung erstellt werden soll, wird i.A. durch die UTM-Benutzerkennung repräsentiert.

Bei Anwendungen oder Clients, die sich nicht explizit mit einer echten Benutzerkennung angemeldet haben, wird der Name der Verbindungs-Benutzerkennung (TS-Anwendungen und UPIC-Clients), der LU6-Sessionname (LU6-Partner) bzw. der OSI-Association-Name (OSI TP-Partner) verwendet.

In UTM-Anwendungen ohne Benutzerkennungen ordnet *open*UTM die Leistungen, die von Terminals, UPIC-Clients oder TS-Anwendungen in Anspruch genommenen wurden, ersatzweise den LTERM-Partnern zu.

Abrechnungsdatei

Alle Informationen, die das UTM-Accounting für die benutzerspezifische Abrechnung der Leistungen sammelt, schreibt *open*UTM in die Abrechnungsdatei.

Die Abrechnungsdatei wird anwendungsspezifisch geführt und vom Administrator der UTM-Anwendung verwaltet, näheres siehe Abschnitt „Auswertung“ auf Seite 160.

Betriebsmittel

Darunter werden folgende Leistungen zusammengefasst:

– technische DV-Leistungen, insbesondere CPU-Verbrauch
– der Aufruf eines bestimmten Programms (Programmgebühr)

Kalkulationsphase

Die Kalkulationsphase dient als Orientierung für den Einsatz des Abrechnungsverfahrens.

In der Kalkulationsphase ermittelt *open*UTM für jedes aufgerufene Teilprogramm den Verbrauch der einzelnen Betriebsmittel und schreibt die Werte als Kalkulationssatz in die Abrechnungsdatei. Näheres ist in Abschnitt „Kalkulationsphase“ auf Seite 155 beschrieben.

Kalkulationssatz

Ein Kalkulationssatz ist ein Satz, den *open*UTM in der Kalkulationsphase pro Teilprogrammlauf in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMK.
Die Datenfelder des Kalkulationssatzes UTMK sind im Anhang auf Seite 206 beschrieben.

Gewicht

Pro Betriebsmittel können Sie ein Gewicht (Faktor) festlegen. Dieses Gewicht gibt an, wie das Betriebsmittel im Vergleich mit anderen Betriebsmitteln zu werten ist. Der Verbrauch eines Betriebsmittels geht dann als Produkt „Gewicht ∗ Betriebsmittelverbrauch“ in die Abrechnung ein. Die Gewichte für die einzelnen Betriebsmittel geben Sie bei der KDCDEF-Generierung in ACCOUNT an, siehe Abschnitt „Variante des Abrechnungsverfahrens festlegen“ auf Seite 156.

Abrechnungsphase

In der Abrechnungsphase ermittelt *open*UTM für jedes Teilprogramm den Betriebsmittelverbrauch. Nach Beendigung eines Teilprogrammlaufs ermittelt *open*UTM anhand der generierten Gewichte und des generierten Festpreises die Summe der Verbrauchswerte.

Folgenden Leistungen werden berücksichtigt:

– CPU-Verbrauch
– erzeugte Ausgabeaufträge für Drucker
– Festpreis für den Aufruf eines Teilprogramms

Das Ergebnis ist eine Anzahl von Verrechnungseinheiten, die auf den benutzerspezifischen Verrechnungseinheitenzähler aufaddiert wird.

*open*UTM schreibt erst dann einen Satz mit dem Inhalt dieses Zählers in die Abrechnungsdatei,
– wenn der Benutzer sich abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist,
– oder wenn die Anwendung normal beendet wird
– oder wenn ein bestimmter Maximalwert überschritten wird. Diesen Maximalwert legen Sie bei der KDCDEF-Generierung mit ACCOUNT ...,MAXUNIT= fest.

Gewichte und Festpreise müssen Sie vor dem Start der Abrechnungsphase in die Generierung der Anwendung einbringen. Dabei können Sie wählen zwischen
– Festpreis-Abrechnung,
– verbrauchsorientierter Abrechnung,
– und einer Kombination der beiden Varianten.

Eine detaillierte Beschreibung der Abrechnungsphase finden Sie in Abschnitt „Abrechnungsphase“ auf Seite 158.

Die Abrechnungsphase des UTM-Accounting kann im laufenden Betrieb der UTM-Anwendung ein- und ausgeschaltet werden.

Abrechnungssatz

Ein Abrechnungssatz ist ein Satz, den *open*UTM in der Abrechnungsphase in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMA.
Die Datenfelder des Abrechnungssatz UTMA sind im Anhang auf Seite 205 beschrieben.

Verrechnungseinheiten

Verrechnungseinheiten sind das Produkt aus Verbrauch und Gewicht des jeweiligen Betriebsmittels. In der UTM-Abrechnung werden nur Verrechnungseinheiten gezählt.

Verrechnungseinheitenzähler

*open*UTM führt in einer UTM-Anwendung pro Benutzer einen Verrechnungseinheitenzähler und akkumuliert dort den Verbrauch der Verrechnungseinheiten pro Benutzer.

Festpreis-Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten in Rechnung gestellt. Diese Anzahl wird bei der Anwendungsgenerierung dem Transaktionscode zugeordnet. Die Gewichte der anderen Betriebsmittel sind dabei Null. Dabei können Sie einzelne Services auch kostenlos anbieten, z.B. Auskunftsfunktionen.

Verbrauchsorientierte Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell ermittelt wird. Die Verbrauchswerte für die Betriebsmittel werden entsprechend der generierten Gewichte gewichtet. Für den Aufruf eines Teilprogramms wird kein Festpreis berechnet.

10.2 Phasen des Accounting

Für die Durchführung des Accounting in UTM-Anwendungen sind folgende Schritte erforderlich:

– Kalkulationsphase
– Abrechnungsverfahren festlegen
– Abrechnungsphase
– Auswertung

10.2.1 Kalkulationsphase

Die Kalkulationsphase liefert Orientierungswerte, mit deren Hilfe Sie die einzelnen Betriebsmittel gewichten und Festpreise für die Inanspruchnahme eines Services festlegen können. Dabei ermittelt *open*UTM für jeden Teilprogrammlauf den Betriebsmittelverbrauch, stellt am Programmlaufende einen Kalkulationssatz vom Satztyp UTMK zusammen und schreibt ihn in die Abrechnungsdatei.

Die Kalkulationsphase kann im laufenden Betrieb jederzeit durch die UTM-Administration ein- und ausgeschaltet werden, z.B. um die generierten Gewichte zu überprüfen und gegebenenfalls bei der Neugenerierung zu aktualisieren.

Sie sollten jedoch beachten, dass *open*UTM bei eingeschalteter Kalkulationsphase nach jedem Teilprogrammlauf einen Satz in die Abrechnungsdatei schreibt. Dadurch wird die Performance der Anwendung belastet.

Kalkulationsphase einschalten

Die Kalkulationsphase können Sie bei der KDCDEF-Generierung oder per Administration einschalten, siehe auch *open*UTM-Handbuch „Anwendungen generieren“ und *open*UTM-Handbuch „Anwendungen administrieren“:

- per KDCDEF-Anweisung ACCOUNT ACC=CALC
- oder per UTM-Administration
  – durch das Kommando KDCAPPL CALC=ON
  – oder über WinAdmin
  – oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Kalkulationsphase ausschalten

Die Kalkulationsphase können Sie nur per UTM-Administration ausschalten:

- durch das Kommando KDCAPPL CALC=OFF
- oder über WinAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten eines Kalkulationssatzes

Ein Kalkulationssatz enthält folgende Daten:

- Name der UTM-Anwendung
- Transaktionscode (TAC) des Teilprogramms
- CPU-Verbrauch im UTM-Prozess in msec
- Länge der Eingabenachricht in byte
- Länge der Ausgabenachricht in byte
- Anzahl Ausgabeaufträge an Drucker
- Verrechnungseinheiten für LTAC-Aufrufe
- UTM-Benutzer, der den Service aufgerufen hat
- Name des LTERM-Partners, über den der Benutzer angemeldet ist
- Realzeit des Teilprogrammlaufs (msec)

Bei den Ausgabenachrichten werden auch die mitgerechnet, die an ein Folgeteilprogramm gerichtet sind (z.B. nach PEND PR).

10.2.2 Variante des Abrechnungsverfahrens festlegen

Zunächst müssen Sie festlegen, ob Sie über Festpreis, über den Verbrauch oder über eine Kombination aus diesen beiden Varianten abrechnen. Ihre Entscheidung hängt davon ab, ob Sie bestimmte Leistungen der Anwendung mit festen Preisen anbieten oder den tatsächlichen Betriebsmittelverbrauch in Rechnung stellen wollen.

Festpreis-Abrechnung

Bei der Festpreis-Abrechnung kostet ein Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten. Orientierungswerte dafür liefert die Kalkulationsphase. Daher ist eine Festpreis-Abrechnung die einfachste Lösung.

Die Anzahl der Verrechnungseinheiten geben Sie bei der KDCDEF-Generierung in der TAC-Anweisung beim Operanden TACUNIT an, siehe *open*UTM-Handbuch „Anwendungen generieren“.

TAC *tacname*,PROGRAM=*progname*,TACUNIT=*anzahl_verrechnungseinheiten*

Für jeden vom Benutzer aufgerufenen Transaktionscode wird der in TACUNIT angegebene Wert auf den benutzerspezifischen Verrechnungseinheitenzähler aufaddiert.

Bei der Festpreis-Abrechnung können Sie einige Services (z.B. Auskunftsfunktionen) auch kostenfrei zur Verfügung stellen. Dazu müssen Sie die zugehörigen Transaktionscodes wie folgt generieren:

```
TAC ... TACUNIT=0
```

Bei Verteilter Verarbeitung gilt Entsprechendes für die Anweisung LTAC und den Operanden LTACUNIT, siehe Abschnitt „Abrechnung bei verteilter Verarbeitung“ auf Seite 161.

Bei einer Festpreis-Abrechnung müssen Sie die Gewichte der Betriebsmittel in der KDCDEF-Anweisung ACCOUNT auf 0 setzen (=Standardwert).

Verbrauchsorientierte Abrechnung

Bei dieser Variante wird dem Benutzer der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell in der Abrechnungsphase ermittelt wird. Für die einzelnen Betriebsmittel müssen Sie Gewichte festlegen. Ein Gewicht ist ein Faktor, mit dem die verbrauchten Einheiten multipliziert werden. Bei der Wahl dieser Gewichte können Sie sich an den Verbrauchsdaten orientieren, die Sie in der Kalkulationsphase ermittelt haben.

Die Gewichte werden anwendungsspezifisch in der KDCDEF-Anweisung ACCOUNT festgelegt, d.h. sie gelten für alle Teilprogrammläufe.

Die Festlegung der Gewichte ist zwangsläufig subjektiv und hängt von der Installationsumgebung ab. Folgende Betriebsmittel können gewichtet werden:

– CPU-Verbrauch (ACCOUNT-Operand CPUUNIT)
– Ein-/Ausgaben auf Hintergrundspeicher (ACCOUNT-Operand IOUNIT)
– Druckerausgaben (ACCOUNT-Operand OUTUNIT)

Näheres siehe *open*UTM-Handbuch „Anwendungen generieren“.

Beispiel für die Generierung dieser Variante

```
ACCOUNT ACC=ON,CPUUNIT=15,IOUNIT=5,OUTUNIT=20
TAC tacname,PROGRAM=progname,TACUNIT=0
TAC ....
```

Pro Aufruf eines Transaktionscodes wird dann die folgende Summe auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

15 ∗ CPU-Verbrauch + 5 ∗ I/O-Verbrauch + 20 ∗ Druckausgaben-Verbrauch

Kombination aus Festpreis- und verbrauchsorientierter Abrechnung

Sie können für Ihre Abrechnung die beiden obigen Varianten auch kombinieren, indem Sie für den Aufruf eines Transaktionscodes einen bestimmten Festpreis festlegen und zusätzlich den Verbrauch der Betriebsmittel (z.B. den CPU-Verbrauch) verrechnen.

In der Abrechnungsphase wird beim Aufruf eines Transaktionscodes die folgende Summe gebildet und auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

TACUNIT (Festpreis für den Aufruf des Teilprogramms)
+ CPUUNIT ∗ CPU-Verbrauch + IOUNIT∗ I/O-Verbrauch
+ OUTUNIT ∗ Druckausgaben-Verbrauch

Beispiel für die Generierung dieser Variante:

```
ACCOUNT ACC=ON,CPUUNIT=15
TAC tacnam1,PROGRAM=progname1,TACUNIT=1
TAC tacnam2,PROGRAM=progname2,TACUNIT=2
:
:
```

10.2.3 Abrechnungsphase

In der Abrechnungsphase ermittelt *open*UTM die pro Teilprogrammlauf verbrauchten Betriebsmittel, berechnet daraus und aus den generierten Gewichten und Festpreisen eine gewichtete Summe. Diese Summe addiert *open*UTM auf den Verrechnungseinheitenzähler des UTM-Benutzers. Der Wert dieses Zählers ist im Abrechnungssatz enthalten, den *open*UTM in die Abrechnungsdatei schreibt.

*open*UTM schreibt immer dann einen Abrechnungssatz, wenn für den Benutzer eine bestimmte Anzahl von Verrechnungseinheiten aufaddiert sind oder wenn sich der Benutzer abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist. Die Anzahl von Verrechnungseinheiten, bei der *open*UTM einen Abrechnungssatz schreibt, legen Sie bei der KDCDEF-Generierung in ACCOUNT MAXUNIT= fest. Dabei müssen Sie Folgendes beachten:

– Den Wert von MAXUNIT sollten Sie nicht zu klein wählen, da ein zu häufiges Schreiben von Abrechnungssätzen die Performance der Anwendung beeinträchtigen könnte.

– Den Wert von MAXUNIT sollten Sie nicht zu groß wählen, da bei einem Abbruch der Anwendung die Verrechnungseinheiten, die noch nicht in die Abrechnungsdatei geschrieben wurden, verloren gehen können (die Abrechnung unterliegt nicht der Transaktionssicherung).

Nachdem der Abrechnungssatz in die Abrechnungsdatei geschrieben wurde, werden der Verrechnungseinheitenzähler und der Zähler für die Anzahl der aufgerufenen TACs auf Null gesetzt.

Abrechnungsphase einschalten

Mit der KDCDEF-Steueranweisung ACCOUNT ACC=ON wird bei der Generierung die Abrechnung für die UTM-Anwendung eingeschaltet.

Zusätzlich kann die Abrechnungsphase auch im laufenden Betrieb durch die UTM-Administration ein- und ausgeschaltet werden

- durch das Kommando KDCAPPL ACCOUNT=ON
- oder durch WinAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Abrechnungsphase ausschalten

Die Abrechnungsphase können Sie nur per Administration ausschalten:

- durch das Kommando KDCAPPL ACCOUNT=OFF
- oder über WinAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten des Abrechnungssatzes

Der Abrechnungssatz ist vom Satztyp UTMA. Er enthält folgende Daten:

- Name der UTM-Anwendung
- UTM-Benutzerkennung
- Zeitpunkt der Anmeldung des Benutzers auf der aktuellen Verbindung
- Wert des Verrechnungseinheitenzählers
- Anzahl der aufgerufenen TACs mit TACUNIT > 0 seit der Anmeldung oder seitdem der letzte Satz geschrieben wurde

Sie können Kalkulationsdaten auch bei laufender Abrechnungsphase erfassen. Damit können Sie die Gewichte jederzeit überprüfen.

10.2.4 Auswertung

Das Ergebnis der Abrechnungsphase sind die Abrechnungssätze in der Abrechnungsdatei. Die Abrechnungsdatei legt *open*UTM im Unterverzeichnis ACCNT des Basisverzeichnisses *filebase* der UTM-Anwendung an. Die Abrechnungsdatei hat den Namen 0001.*pid*. Dabei ist *pid* die Prozess-ID des Logging-Prozesses der UTM-Anwendung, in dem das Programm *utmlog* abläuft.

In die Datei 0001.*pid* schreibt *open*UTM sowohl die Abrechnungssätze als auch die Kalkulationssätze.

Nach dem Aus- und Wiedereinschalten der Abrechnungsphase schreibt *open*UTM in derselben Datei weiter. Die vor dem Ausschalten erfassten Kalkulations- und Abrechnungsätze werden nicht überschrieben.

Die Datei können Sie selbst auswerten.

Der Aufbau der Abrechnungssätze ist im Anhang auf Seite 205 beschrieben.

10.2.5 Fehlersituationen

Kann das Accounting auf Grund eines Fehlers einen Abrechnungssatz und/oder einen Kalkulationssatz nicht schreiben, z.B. weil nicht genügend Platz auf der Platte ist, erzeugt *open*UTM die Meldung K079 und beendet die Kalkulations- und/oder Abrechnungsphase. Ein Insert der Meldung K079 gibt die Ursache des Fehlers an. Die Anwendung läuft weiter.

Nach Behebung des Fehlers kann die Kalkulations- und/oder Abrechnungsphase durch die UTM-Administration (z.B. mit dem Administrationskommando KDCAPPL) wieder eingeschaltet werden.

10.3 Abrechnung bei verteilter Verarbeitung

Die lokale Anwendung (Auftraggeber) nutzt Leistungen, die Teilprogramme in fernen Partneranwendungen (Auftragnehmer) für sie erbringen. In diesem Fall kann in der Auftraggeber-Anwendung der dabei anfallende Betriebsmittelverbrauch als Festpreis in Rechnung gestellt werden. Dazu ordnen Sie den LTACs in der Auftraggeber-Anwendung Verrechnungseinheiten als Festpreise zu. LTACs sind die Transaktionscodes, die in der Auftraggeber-Anwendung für einen Vorgang in einer Auftragnehmer-Anwendung definiert werden.

Näheres siehe *open*UTM-Handbuch „Anwendungen generieren“, Anweisung LTAC, Operand LTACUNIT.

Kalkulationsphase (Festlegen der Festpreise)

In der **Auftragnehmer-Anwendung** wird in der Kalkulationsphase der mittlere Betriebsmittelverbrauch der Teilprogramme ermittelt, die Services für die Auftraggeber-Anwendung erbringen. Anhand der hier ermittelten Verbrauchswerte können Sie die Festpreise festlegen, die den Benutzern der LTACs in der Auftraggeber-Anwendung berechnet werden sollen.

In der **Auftraggeber-Anwendung** zählt *open*UTM in einem Feld des Kalkulationssatzes die Verrechnungseinheiten, die bei den Aufrufen von LTACs anfallen.

Abrechnungsphase

In der **Auftragnehmer-Anwendung** werden alle Verbrauchswerte, die bei der Bearbeitung von Aufträgen für eine Auftraggeber-Anwendung anfallen, wie folgt zugeordnet:

– Bei LU6.1 den Sessions (LSES) zum Auftraggeber

– Bei OSI TP den Associations (OSI-LPAP ... ,ASSOCIATION-NAME=), falls sich der OSI-TP-Auftraggeber nicht unter einer echten Benutzerkennung angemeldet hat.

Die erbrachten Leistungen werden also der Auftraggeber-Anwendung insgesamt in Rechnung gestellt. Die Aufwände für die einzelnen Benutzer der Auftraggeber-Anwendung können nicht ermittelt werden.

In der **Auftraggeber-Anwendung** addiert *open*UTM beim Aufruf eines LTACs die Verrechnungseinheiten auf den Verrechungseinheitenzähler des Benutzers auf, die in der KDC-DEF-Generierung in der LTAC-Anweisung angegeben wurden.

10.4 Einschränkungen

Bei der Nutzung des UTM-Accounting ist Folgendes zu beachten:

- Das Schreiben von Abrechnungsinformationen unterliegt nicht der Transaktionssicherung, deshalb können beim Abbruch einer Anwendung Verrechnungseinheiten verloren gehen. Der Maximalwert pro Benutzer kann per Generierung begrenzt werden.
- Für Anwendungen mit verteilter Verarbeitung wird in der Kalkulationsphase jeder LTAC-Aufruf mitgezählt. Es wird nicht berücksichtigt, ob nach der PEND-Verarbeitung eine Session belegt werden konnte oder nicht.
- Die Erfassung des Betriebsmittelverbrauchs beginnt vor dem Start eines Teilprogramms, sie endet in der Verarbeitung des PEND-Aufrufs. Die übrige Verarbeitungsleistung (Grundverbrauch) der UTM-Tasks wird den Benutzern nicht in Rechnung gestellt.
- Das Rücksetzen einer Transaktion hat folgende Auswirkungen: Alle Werte bis auf CPU werden zurückgesetzt. Da *open*UTM die Verbrauchswerte in der PEND-Verarbeitung aufsummiert, kann eine Rücksetzaktion Verbrauchswerte nur dann zurücksetzen, wenn sie im aktuellen Teilprogrammlauf entstehen.
- Sind seit dem letzten Start der Anwendung für den Benutzer nur Asynchronaufträge verarbeitet worden, steht im Abrechnungssatz als Zeitpunkt der Anmeldung an die Anwendung der Wert Null.
- Für den Event-Exit VORGANG wird der Betriebsmittelverbrauch nur am Anfang des Vorgangs erfasst.
- Für den Event-Service BADTACS kann in der Abrechnungsphase kein Teilprogramm-Gewicht berücksichtigt werden.

11 Leistungskontrolle mit KDCMON

Die Performance einer UTM-Anwendung wird von verschiedenen Faktoren beeinflusst. Die Einflussfaktoren liegen zum einen im System-Umfeld einer UTM-Anwendung (Ausbau des Arbeitsspeichers, Leistungsfähigkeit der Peripherie) und zum anderen in der UTM-Anwendung selbst (Konfiguration der Anwendung und Aufbau der Teilprogramme). Im laufenden Betrieb einer Anwendung sollten Sie regelmäßig Leistungskontrollen durchführen, damit Sie rechtzeitig Hinweise auf Leistungsengpässe erhalten. Für UTM-Anwendungen stehen Ihnen die folgenden Instrumente zur Leistungskontrolle zur Verfügung:

– UTM-Messmonitor KDCMON mit dem Auswertungstool KDCEVAL

– Informationsservices der UTM-Administration

UTM-Messmonitor KDCMON

Für UTM-Anwender steht der UTM-Messmonitor KDCMON zur Verfügung. KDCMON ist eine in *open*UTM integrierte Funktion und zeichnet Informationen über den Ablauf von UTM-Anwendungen und Anwenderteilprogrammen auf. Zeichnen sich Leistungsengpässe ab, können Sie mit KDCMON Daten erfassen. Die gesammelten Daten müssen Sie mit dem Tool KDCEVAL auswerten. Anhand dieser Auswertung können Sie eine detaillierte Analyse durchführen. Siehe ab Seite 169.

KDCMON ist damit ein wichtiges Werkzeug zur Beurteilung der Performance in einer UTM-Anwendung. KDCMON kann z.B. zur detaillierten Leistungsuntersuchung eingesetzt werden, wenn Messungen mit Hilfe der UTM-Administration auf Leistungsengpässe hinweisen.

Informationsservices der UTM-Administration

Detaillierte Informationen zur Diagnose von Leistungsengpässen können Sie auch per UTM-Administration abfragen, z.B. über das Administrationskommando KDCINF oder über das Administrationsprogramm WinAdmin. Das Kommando KDCINF STATISTICS liefert Daten über die Auslastung einzelner ausgewählter Komponenten Ihrer UTM-Anwendung, wie beispielsweise der Clients. Mit dem Kommando KDCINF STATISTICS können Sie gleichzeitig allgemeine Statistikinformationen über die Auslastung der gesamten Anwendung abfragen und Kenngrößen für die Leistungskontrolle und die Beurteilung der Performance Ihrer UTM-Anwendungen im laufenden Betrieb ermitteln, wie beispielsweise

Belegung des Pagepools, Anzahl der aktuell angemeldeten Benutzer, Anzahl der durchgeführten Dialog- oder Asynchron-Transaktionen pro Sekunde, offene Dialog- und Asynchronvorgänge usw.. Siehe auch im *open*UTM-Handbuch „Anwendungen administrieren“.

Wenn Sie die UTM-Anwendung mit dem grafischen Administrationsarbeitsplatz WinAdmin administrieren, können die Statistikdaten auch grafisch dargestellt werden.

11.1 Erfassung starten und beenden

Die Datenerfassung können Sie mit dem folgenden Administrations-Kommando ein- und ausschalten:

```
KDCDIAG KDCMON={ ON | OFF }
```

Diese Administrationsfunktion steht auch an der Programmschnittstelle KDCADMI und beim Administrationsprogramm WinAdmin zur Verfügung.

Der UTM-Administrator kann jederzeit mit Hilfe des Kommandos `KDCINF SYSPARM` feststellen, ob Daten erfasst werden oder nicht.

Wenn *open*UTM beim Einschalten feststellt, dass die KDCMON-Funktion nicht verfügbar ist, dann wird folgende Meldung mit dem Standard-Ziel SYSLOG ausgeben:

```
K080 KDCMON-Funktion nicht verfuegbar
```

Mögliche Ursache: Die Kommunikation mit dem Log-Prozess ist gestört (nur bei UNIX).

Wenn *open*UTM bei laufender Erfassung feststellt, dass die KDCMON-Funktion nicht mehr verfügbar ist, dann schaltet *open*UTM die Datenerfassung aus und protokolliert dies ebenfalls mit der Meldung K080.

In UNIX und Windows werden beim Erfassen folgende Dateien erzeugt:

UNIX Für jedes Messintervall erzeugt *open*UTM eine Datei mit folgendem Namen:

filebase`/KDCMON/`*nnnn.pid*

Dabei ist *nnnn* die laufende Nummer des Messintervalls, beginnend bei 0001 nach dem Anwendungsstart, und *pid* die Prozess-ID des utmlog-Prozesses, der zum Anwendungslauf gehört.

Windows *open*UTM erzeugt eine Datei mit dem festen Namen *filebase*`\KDCMON\0001`.

Diese Datei bleibt vom ersten Kommando KDCDIAG KDCMON=ON bis zum Ende des Anwendungslaufs geöffnet. Sie können diese Datei kopieren, nachdem Sie das Kommando KDCDIAG KDCMON=OFF eingegeben haben.

Bitte beachten Sie, dass die Datei nach dem nächsten Kommando KDCDIAG KDCMON=ON oder nach dem nächsten Start der Anwendung wieder leer ist.

11.2 Daten auswerten mit KDCEVAL

Mit dem Tool KDCEVAL werden die mit KDCMON aufgezeichneten Daten ausgewertet. In einem Auswertungslauf können nur die Daten *einer* Anwendung ausgewertet werden. KDCEVAL benötigt zur Steuerung einige Parameter, die Sie nach dem Starten von KDCEVAL eingeben müssen.

Starten von KDCEVAL

Vor dem Starten von KDCEVAL müssen Sie die Datei, die Sie auswerten möchten (*nnnn.pid* im Verzeichnis `KDCMON`, s.o.), in das aktuelle Verzeichnis kopieren und unter dem Namen `evalin` abspeichern.

Danach rufen Sie KDCEVAL wie folgt auf:

utm-dateiverzeichnis`/ex/kdceval` (UNIX) bzw.

utm-dateiverzeichnis`\ex\kdceval` (Windows)

Nach dem Start des Auswertungsprogramms im Dialog fordert KDCEVAL mit der folgenden Meldung die Eingabe von Steuerparametern an:

```
PLEASE ENTER COMMANDS OR 'HELP' OR 'END'
```

Steuerparameter von KDCEVAL

Das Programm liest die Parameter von *stdin*. Die einzelnen Kommandos, mit denen Sie die Auswertung steuern können, haben folgendes Format:

APPLINAME appliname
: Name der Anwendung, für die die Auswertung durchgeführt werden soll.

TIME FROM={ t1 | START }, TO={ t2 | STOP }
: Zeitangabe zur Definition des Auswertungsintervalls.

  FROM=t1
  : Startzeitpunkt der Auswertung in Sekunden.
    Die Zeitangabe erfolgt relativ zum Einschalten der Erfassung (z.B. mit dem Kommando KDCDIAG).

  FROM=START
  : Es wird vom Anfang der Datei an ausgewertet.

  TO=t2 Endzeitpunkt der Auswertung in Sekunden.
  : Die Zeitangabe erfolgt relativ zum Einschalten der Erfassung (z.B. mit dem Kommando KDCDIAG).

TO=STOP
Es wird bis zum Dateiende ausgewertet.

Für *t1* und *t2* gilt:

Minimalwert: 0
Maximalwert: 99999999

LIST { (liste$_1$, liste$_2$,...,liste$_n$ [,TABLE]) | (STD [,TABLE]) | (ALL [,TABLE]) }

liste$_1$, liste$_2$,...,liste$_n$
Namen der Einzellisten, die ausgewertet werden sollen. Die Namen, die Sie hier angeben können, finden Sie auf Seite 170. Die Listen TRACE und TRACE2 dürfen nicht zusammen angegeben werden.

STD Diese Auswertung umfasst die Listen TASKS, SUMM, TIMES und TCLASS.

ALL Die Auswertung umfasst alle Listen außer TRACE und TRACE2.

Werden ALL oder STD ohne TABLE angegeben, dann können die runden Klammern weggelassen werden.

TABLE
Wird zusätzlich TABLE angegeben, dann werden die Listen in einem Tabellenformat erzeugt, das auf einem PC mit Excel oder einem anderen Tabellenkalkulator bearbeitet werden kann, siehe Seite 168. TABLE wirkt nur auf die Einzellisten TASKS, TIMES, TCLASS, TACCL, TACPT und TACLIST.

OPTION **D**ECIMAL-**S**EPARATOR={ **C**OMMA | **P**OINT }
Definiert das Dezimaltrennzeichen.

DECIMAL-SEPARATOR=COMMA
Als Dezimaltrennzeichen wird das Komma verwendet.

DECIMAL-SEPARATOR=POINT
Als Dezimaltrennzeichen wird der Punkt verwendet, Standardwert.

END Mit diesem Kommando wird die Parametereingabe beendet.

Bei Auswertungen im Dialog kann auch das Kommando HELP eingegeben werden. Es werden dann die Syntax der Kommandos und die möglichen Listennamen ausgegeben.

Fehler und Meldungen

- Fehlt eines der Kommandos APPLINAME, TIME oder LIST, so wird die Auswertung mit folgender Fehlermeldung abgebrochen:

  ```
  MANDATORY COMMAND MISSING
  ```

- Bei einem Syntaxfehler erscheint die folgende Meldung und das fehlerhafte Kommando wird angezeigt:

  ```
  ERROR IN COMMAND
  ```

- Sind die Zeitangaben *t1* und *t2* inkonsistent, so wird folgende Meldung ausgegeben:

  ```
  KDCEVAL: WRONG TIME INPUT
  ```

- Werden in der Datei keine Sätze für die Anwendung gefunden oder sind für das Auswertungsintervall keine Daten vorhanden, so wird eine der folgenden Meldungen ausgegeben:

  ```
  NO EVALUATION : NO RECORD WITH APPLINAME FOUND
  ```

  oder

  ```
  NO EVALUATION : NO RECORD IN TIME_INTERVAL
  ```

- Bei einem DMS-Fehler werden folgende Meldungen ausgegeben:
  - Wenn KDCEVAL die Datei `evalin` nicht gefunden hat:

    ```
    KDCEVAL: NO KDCMON FILE
    KDCEVAL: NO EVALUATION
    ```

  - Wenn die Datei `evalin` nicht durch KDCMON erzeugt wurde:

    ```
    NO EVALUATION: NO VALID KDCMON FILE
    ```

- Versionsprüfung:

  Die Auswertung der KDCMON-Daten durch KDCEVAL ist nur möglich, wenn KDCEVAL die gleiche UTM-Version wie der UTM-Systemcode hat. KDCEVAL prüft die Version der KDCMON-Daten. Erkennt KDCEVAL eine unzulässige Version, bricht KDCEVAL die Auswertung mit der folgenden Meldung ab:

  ```
  NO EVALUATION: INPUT FILE FROM INVALID UTM VERSION
  ```

Ergebnis der Auswertung mit KDCEVAL

KDCEVAL schreibt die Auswertung in die Datei

`kdcmon.`*`appliname`*

Diese Datei wird im aktuellen Verzeichnis abgelegt.

11.3 Auswertungsdaten auf dem PC bearbeiten

Wenn Sie bei KDCEVAL im Steuerparameter LIST zusätzlich zu den Listennamen den Operanden TABLE angeben, dann werden diese Listen in Tabellenform erzeugt. Diese Art der Aufbereitung ist nur für Listen TASKS, TIMES, TCLASS, TACCL, TACPT, und TACLIST möglich.

Diese so erzeugten Listen können Sie auf dem PC mit einem Tabellenkalkulator wie z.B. Excel bearbeiten und grafisch aufbereiten. Für Excel wird dazu das Makro `kdceval.xls` ausgeliefert.

Dazu gehen Sie wie folgt vor:

1. Übertragen Sie die von KDCEVAL erzeugte Listen-Datei sowie das Makro `kdceval.xls` auf einen PC.

   Das Makro verlangt, dass die auszuwertende Datei die Endung `.txt` besitzt!

2. Rufen Sie das Makro `kdceval.xls` auf und lesen Sie die Listen-Datei in Excel ein. Excel erzeugt dann für jede Liste ein eigenes Tabellenblatt sowie ein Zusatzblatt mit Übersichtsinformationen (Summary-Blatt).

3. Bearbeiten Sie die einzelnen Listen nach Ihren Wünschen, indem Sie z.B. eine Liste sortieren und anschließend in ein Kurven- oder Balkendiagramm umwandeln.

11.4 Auswertungslisten

Jede Auswertungsliste besteht aus

– einer Überschrift, die den Namen der Auswertungsliste enthält
– einem Kopf, der für alle Listen identisch ist
– der spezifischen Auswertungsliste

Der Listenkopf hat folgenden Aufbau:

```
NAME OF APPLICATION   : appli      DATE   weekday month dd hh:mm:ss year
COMMENCEMENT TIME     : sss SEC.   KDCEVAL VERSION     : V5.2Ann
END TIME              : sss SEC.   UTM VERSION         : V5.2Ann
SYSTEM INFORMATION    : system name and bitmode
```

mit folgender Bedeutung der Felder (soweit nicht selbsterklärend):

NAME OF APPLICATION
: Name der Anwendung

DATE
: Datum der Datenerfassung mit KDCMON

COMMENCEMENT TIME
: Anfangszeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt der Datenerfassung)

END TIME
: Endzeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt der Datenerfassung)

SYSTEM INFORMATION
: Name und Betriebssystemversion des Rechners sowie Ablaufmodus (Bitmode: 32 oder Bitmode: 64)

Bei der TRACE- und TRACE2-Liste enthält END TIME den Wert 999999, wenn man die ganze Datei auswertet (Parameter TIME FROM=START,TO=STOP).

Bei den Bearbeitungszeiten handelt es sich stets um die ELAPSED TIME (Realzeit).

Es sind folgende Einzel-Auswertungen und Kombinationen dieser Auswertungen möglich:

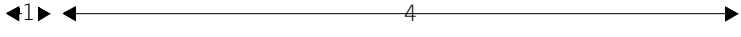
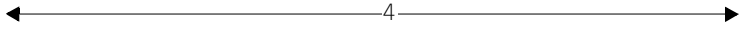
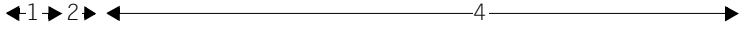
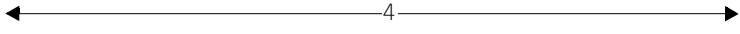
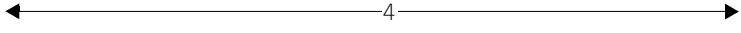
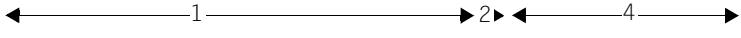
TASKS	UTILIZATION OF THE UTM TASKS
SUMM	TRANSACTION EVALUATION
TIMES	DISTRIBUTION OF PROCESSING TIMES
KCOP	KDCS CALLS STATISTIC
WAIT	WAITING TIMES
TCLASS	EVALUATION OF THE TAC CLASSES
TACCL	TAC SPECIFIC TAC CLASS EVALUATION
TACPT	TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES
TACLIST	TAC SPECIFIC STATISTICS
TRACE	TASK SPECIFIC TRACES
TRACE2	TASK PERFORMANCE TRACES

Im Folgenden werden die einzelnen Auswertungslisten beschrieben.

TASKS: UTILIZATION OF THE UTM TASKS

In dieser Liste wird ein Überblick über die Auslastung der Prozesse der Anwendung gegeben. Außerdem wird der CPU-Verbrauch für jeden einzelnen UTM-Prozess aufsummiert und für alle Prozesse der Anwendung angezeigt.

```
1   Program
2   System code
4   Bourse Wait
```

PID	START TIME	TASK UTILIZATION
1564	12:14:07.840	
1137	12:08:29:190	
0882	12:00:23.210	
0730	11:58:14.610	
0676	11:57:02.560	
0115	11:55:19.290	

PID	CPU-time	Program	System	Bourse Wait
1564	40010	40330	12190	1283200
1137	460	450	4010	1623370
0882	157830	161180	56830	2083810
0730	19490	19270	1740	2240400
0676	240	10	230	2274550
0115	21920	1467390	61240	949110
Summ	239950			

In der Liste bedeuten:

PID	Prozess ID des UTM-Prozesses.
START TIME	Zeit des ersten Satzes dieses Prozesses (absolut).
program	Zeitanteil des Anwendungsprogramms im UTM-Prozess.
system code	Zeitanteil des UTM-Systemcodes.
bourse wait	Zeitanteil, den der Prozess auf neue Aufträge an der Auftragswarteschlange gewartet hat.

Für die Auswertung TASKS ist ein Herabsetzen der Prozess-Anzahl während des Auswertungsintervalls zu vermeiden, denn es führt zu falschen Ergebnissen. In diesem Fall sollten Sie weitere Auswertungsintervalle verwenden.

SUMM: TRANSACTION EVALUATION

In dieser Liste wird ein Überblick über die Vorgänge und Transaktionen für den Auswertungszeitraum gegeben. Es werden nur solche Transaktionen berücksichtigt, die vollständig im Auswertungszeitraum liegen. Zusätzlich gibt das Auswertungstool KDCEVAL den CPU-Verbrauch von allen Teilprogrammläufen aus, die im Auswertungsintervall beendet wurden. Die Liste hat folgendes Format:

```
COUNT OF TRANSACTIONS                          :   240 1)
COUNT OF SERVICES                              :    80 2)
COUNT OF DIALOG STEPS                          :   290
NUMBER OF DIALOG STEPS PER SECOND              :     4
                                               :
TOTAL CPU-TIME USED IN MSEC                    :  9150 3)
```

1) Die Transaktion KDCDIAG zum Ein- und Ausschalten des Messmonitors wird nicht gezählt.

2) SERVICE = Vorgang

3) In dieser Zeile wird die Summe des CPU-Verbrauchs der einzelnen Teilprogrammläufe ausgegeben. Darin ist auch der Verbrauch im UTM- und Betriebssystemcode enthalten, soweit er innerhalb der Teilprogrammläufe anfällt, sowie die Start- und Endeverarbeitung in *open*UTM. Andere Aktionen der UTM-Prozesse, die nicht direkt zu Teilprogrammläufen gehören, sind nicht enthalten.

TIMES: DISTRIBUTION OF PROCESSING TIMES

In dieser Liste wird in tabellarischer Form eine grobe Verteilung der Bearbeitungszeiten für die Teilprogramme ausgegeben. In diesen Zeiten ist die Wartezeit vor der Bearbeitung durch *open*UTM nicht enthalten.

Die Liste hat folgendes Format:

PROCESSING TIMES (MSEC)	NUMBER	PERCENT
0 - 100	76	58.91
101 - 200	45	34.88
201 - 500	7	5.42
501 - 1000	1	0.77
1001 - 2000	0	0.00
2001 - 5000	0	0.00
5001 - 10000	0	0.00
10001 - 20000	0	0.00
20001 - 50000	0	0.00
50001 - 100000	0	0.00
> 100000	0	0.00

In dieser Liste ist die Anzahl der vollständigen Teilprogrammläufe und der prozentuale Anteil für die jeweilige Zeitklasse angegeben.

KCOP: UTM CALLS STATISTIC

In dieser Tabelle ist für die UTM-Aufrufe angegeben, wie oft sie im Auswertungszeitraum aufgetreten sind.

Unter *others* sind Aufrufe aufgeführt, die nicht in der Liste der von *open*UTM unterstützten Aufrufe enthalten sind.

In dieser Liste sind Aufrufe enthalten, die *open*UTM für interne Bearbeitungen aufruft und die dem Anwender nicht zur Verfügung stehen:

CONT	Aufruf nach der Formatierung oder einer internen Datenbankkommunikation
ADMI	UTM-Administrationsaktion
WAIT	Ende der Bearbeitung eines Programmlaufes
NOOP	Pufferbereich der MESSAREA muss geleert werden

Die KCOP-Liste hat folgendes Format:

OP	OM	NUMBER
INIT		7662
INIT	PU	0
INIT	MD	0
CONT		1857
CTRL	AB	0
CTRL	PE	0
CTRL	PR	0
CTRL	EC	0
APRO	AM	0
APRO	DM	0
APRO	IN	0
MCOM	BC	0
MCOM	EC	0
MGET		7331
MPUT	NT	8
MPUT	NE	5753
MPUT	PM	0
MPUT	RM	4
MPUT	EM	0
MPUT	HM	0
MPUT	ES	0

OP	OM	NUMBER
SREL	GB	0
GTDA		0
PTDA		0
UNLK	DA	0
UNLK	GB	0
UNLK	US	0
LPUT		0
INFO	LO	0
INFO	CD	0
INFO	CK	0
INFO	DT	0
INFO	SI	0
INFO	FH	0
INFO	GN	0
INFO	PC	0
DADM	RQ	0
DADM	UI	0
DADM	CS	0
DADM	DL	0
DADM	DA	0
PADM	OK	0

OP	OM	NUMBER
MPUT	GC	0
FGET		0
FPUT	NT	0
FPUT	NE	0
FPUT	RP	0
DPUT	NI	0
DPUT	NT	0
DPUT	NE	0
DPUT	+I	0
DPUT	-I	0
DPUT	+T	0
DPUT	-T	0
DPUT	QT	0
DPUT	QE	0
DPUT	QI	0
DPUT	RP	0
DGET	FT	0
DGET	NT	0
DGET	BF/BN	0
DGET	PF/PN	0
QCRE	NN	0
QCRE	WN	0
QREL	RL	0
SPUT	DL/MS/ES	42284
SPUT	GB	0
SPUT	US	0
SGET	KP	2917
SGET	GB	0
SGET	RL	0
SGET	US	0
SREL	LB	0
OTHERS		0

OP	OM	NUMBER
PADM	PR	0
PADM	AT	0
PADM	AC	0
PADM	CA	0
PADM	CS	0
PADM	AI	0
PADM	PI	0
SIGN	ST	0
SIGN	ON	0
SIGN	CP	0
SIGN	CK	0
SIGN	OF	0
SIGN	OB	0
SIGN	CL	0
SMSG		0
NOOP		1111
WAIT		5762
ADMI		80
RSET		132
PGWT	CM	0
PGWT	RB	0
PGWT	KP	0
PGWT	PR	0
PEND	SP	1900
PEND	RE	65
PEND	FI	323
PEND	ER	0
PEND	FR	0
PEND	KP	5370
PEND	RS	4
PEND	PS	0
PEND	FC	0
PEND	PA/PR	0

WAIT: WAITING TIMES

Zur Ermittlung von Stausituationen fügt *open*UTM bei eingeschaltetem KDCMON in regelmäßigen Zeitabständen Pseudoaufträge in die Auftragswarteschlange ein. Mit Hilfe des Zeitpunktes, zu dem der Auftrag eingefügt wurde (Zeitstempel absolut) und dem Bearbeitungszeitpunkt kann die Wartezeit der Aufträge in der UTM-Warteschlange ermittelt werden. Der Zeitabstand zwischen den einzelnen Pseudoaufträgen beträgt etwa 10 Sekunden. Er kann aber auch größer sein, da die Initiierung dieser Pseudoaufträge an bestimmte UTM-Ereignisse gekoppelt ist.

In der WAIT-Liste wird Folgendes protokolliert:

- In der Spalte WAITING TIME wird die ermittelte Wartezeit für jeden Pseudoauftrag in Sekunden ausgegeben.
- Für diese Wartezeiten berechnet das Auswertungstool KDCEVAL zusätzlich den Maximal-, Minimal- und Mittelwert in Sekunden und zeigt diese Werte unter UTM WAITING TIMES an.
- In der Spalte NUMBER OF TASKS wird die Anzahl der Prozesse, die zu diesem Zeitpunkt in der Anwendung zur Verfügung standen, ausgegeben.

Wenn die Wartezeit zu lang wird, sollte die Anzahl der UTM-Prozesse erhöht werden.

Die WAIT-Liste hat folgendes Format:

TIME STAMP	WAITING TIME	NUMBER OF TASKS
10:33:35.742	0.018	1
10:33:45.781	0.008	2
10:33:55.841	0.007	3
10:33:66.027	0.008	3
10:33:76:087	0.008	3
10:33:86:112	0.007	3
10:33:96:123	0.007	3

```
                        UTM WAITING TIMES:

TIME STAMP :  10:33:35.742    WAITING TIME MAXIMUM :        0.018
TIME STAMP :  10:33:41.740    WAITING TIME MINIMUM :        0.007
NUMBER OF ENTRIES:       7    WAITING TIME AVERAGE :        0.009
```

TCLASS: EVALUATION OF THE TAC CLASSES

Die TCLASS-Liste enthält in tabellarischer Form einen Überblick über die Auftragsbearbeitung von TACs der einzelnen TAC-Klassen (1 bis 16). In der Auswertung sind in der TAC-Klasse 0 alle die Dialog-TACs zusammengefasst, denen bei der Generierung mit KDCDEF keine TAC-Klasse zugeordnet wurde.

Bei der UTM-Generierung kann der Anwender festlegen, wieviele Prozesse zu einem Zeitpunkt maximal für eine TAC-Klasse arbeiten dürfen. Ist diese Anzahl erreicht, so werden weitere Aufträge in eine TAC-Klassen-spezifische Warteschlange eingereiht.

TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT			AVERAGE WAIT TIME (IN MSEC)	MAXIMUM WAIT TIME (IN MSEC)	MINIMUM WAIT TIME (IN MSEC)
		NUMBER CALLS	WAIT TIME=0	WAIT TIME>0			
0	21	30.87					
1	2	2.94	100.00	0.00	0	0	0
2	4	5.88	75.00	25.00	3000	5000	1000
3	3	4.41	100.00	0.00	0	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
8	5	7.35	80.00	20.00	5000	8000	2000
9	2	2.94	90.00	10.00	4000		
10	3	4.41	100.00	0.00	0	0	0
11	4	5.88	75.00	25.00	3000	5000	1000
12	3	4.41	100.00	0.00	0	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
16	5	7.35	80.00	20.00	5000	8000	2000
			90.00	10.00	4000		

41 DIALOG TACS WERE CALLED
27 ASYNCHRONOUS TACS WERE CALLED

Die TCLASS-Liste enthält folgende Angaben:

- In der Spalte TAC-CLASS wird für eine TAC-Klasse die Anzahl der TAC-Aufrufe im Auswertungszeitraum angegeben.

- Die Spalte DISTRIBUTION IN PERCENT enthält Prozentwerte.
  Die Unterspalte NUMBER CALLS gibt den prozentualen Anteil der Aufrufe einer TAC-Klasse zur Anzahl aller Dialog-TAC-Aufrufe an. Die nächsten beiden Spalten enthalten eine prozentuale Aufteilung der Aufrufe dieser TAC-Klasse in
  - Aufrufe, die sofort bearbeitet wurden (WAITTIME=0) und
  - Aufrufe, die in eine TAC-klassenspezifische Warteschlange eingereiht werden mussten (WAITTIME>0)
- In der Spalte AVERAGE WAIT TIME wird die mittlere Wartezeit pro TAC-Klasse angezeigt.
- Die Werte in den Spalten AVERAGE / MINIMUM / MAXIMUM WAIT TIME beziehen sich auf die Aufträge, die *open*UTM vorübergehend in eine TAC-klassenspezifische Warteschlange verdrängt hat. Es wird die mittlere, minimale bzw. maximale Wartezeit eines Auftrags pro TAC-Klasse angezeigt.

i Die mittlere Wartezeit von Aufträgen pro TAC-Klasse kann im laufenden Betrieb einer Anwendung auch mit dem Administrationskommando KDCINF TACCLASS oder mit der entsprechenden Funktion bei WinAdmin oder KDCADMI abgefragt werden.

Wartezeit bei Dialogaufträgen

Bei Dialogaufträgen ist die Wartezeit die Zeitspanne zwischen der Entgegennahme des Auftrags durch die Anwendung (Abholen des Auftrags von der Warteschlange der Anwendung) und dem Start des Teilprogramms. Es kann auch zwischen einzelnen Teilprogrammen zur Verdrängung kommen.

Wartezeit bei Asynchronaufträgen

*open*UTM erfasst auch die Wartezeit von Asynchronaufträgen, die sich wie folgt definiert:

Asynchronauftrag	**Definition „Wartezeit“**
Eingabe Asynchron-TAC	Zeitspanne zwischen der Entgegennahme des Auftrags durch *open*UTM und dem Starten des Asynchronvorgangs
FPUT-Aufruf im Teilprogramm	Zeitspanne zwischen dem Ende der Transaktion, in der der FPUT-Auftrag ausgeführt wurde, und dem Starten des Asynchronvorgangs
DPUT-Aufruf im Teilprogramm	Zeitspanne zwischen der Umwandlung des DPUT in einen FPUT und dem Starten des Asynchronvorgangs

Wurde der Asynchronauftrag nicht im aktuellen Anwendungslauf erzeugt, so wird als asynchrone Wartezeit immer die Zeitdifferenz zwischen dem Start der Anwendung und dem Start des Asynchronauftrags genommen.

TACCL: TAC SPECIFIC TAC CLASS EVALUATION

In der Liste TACCL sind die gleichen Informationen wie in der TCLASS-Liste aufgeführt, nur aufgegliedert auf die einzelnen Transaktionscodes. Es sind alle TACs aufgeführt, die im Auswertungszeitraum aufgerufen wurden. Die TACs sind in der Reihenfolge des ersten Auftretens aufgeführt. Die Bedeutung der einzelnen Spalten kann der Beschreibung des TCLASS-Listenformates entnommen werden.

TAC	TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT			AVERAGE WAIT TIME (IN MSEC)
			NUMBER CALLS	WAIT TIME=0	WAIT TIME> 0	
TAC1	0	10	22.50			
TAC2	5	5	11.25	60.00	40.00	1000
TAC3	2	1	2.50	0.00	100.00	3256
TAC4	13	12	23.40	40.00	60.00	2000
TAC5	15	17	26.20	55.00	45.00	4000
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Für TACs der TAC-Klasse 0 werden keine Angaben für WAIT TIME eingetragen.

TACPT: TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES

In dieser Tabelle sind für alle im Auswertungszeitraum bearbeiteten TACs die minimale (MIN), die maximale (MAX) und die mittlere (MEAN) Bearbeitungszeit aufgeführt. Es werden nur solche TACs aufgenommen, deren Start- und Beendigungszeitpunkt innerhalb des Auswertungszeitraums liegen. Die Liste hat folgendes Format:

TAC	PROCESSING TIME PER TAC (IN MSEC)		
	MEAN	MIN	MAX
TAC1	150000	125000	175000
TAC2	35000	19000	45000
TAC3	24500	20000	29000

Die Tabelle ist nach mittleren Bearbeitungszeiten fallend sortiert. Angezeigt werden nur TACs mit einer mittleren Bearbeitungszeit > 0.

TACLIST: TAC SPECIFIC STATISTICS

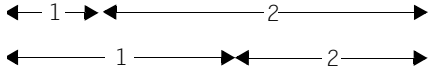
In dieser Liste werden die folgenden TAC-spezifischen Informationen erfasst:

- die durchschnittliche KB-Größe (Spalte AVERAGE SIZE OF KB)
- die Aufteilung der Bearbeitungszeit in

  1: program
  2: system code

Die Liste hat folgendes Format:

TAC	NUMBER CALLS	AVERAGE SIZE OF KB
TAC1	5	256
TAC2	18	1024
.	.	.
.	.	.



Die Liste ist nicht sortiert, die TACs sind in der Reihenfolge aufgelistet, in der sie in der Datei zuerst auftreten.

Es werden nur die TACs berücksichtigt, die einen vollständigen Börsenzyklus hatten.

TRACE: TASK SPECIFIC TRACES

Zur genaueren Analyse des Ablaufs einer UTM-Anwendung können TRACE-Listen erstellt werden. In diesen Listen werden in zeitlicher Reihenfolge für die einzelnen UTM-Prozesse alle UTM-Aufrufe aufgelistet.

Es wird immer nur eine TRACE-Liste mit den Daten der ersten 6 Prozesse erzeugt. Existieren für den Auswertungszeitraum Daten von mehr als 6 Prozessen, dann sollte zur Auswertung die Tabelle TRACE2 verwendet werden.

Die Liste ist in zeitlicher Reihenfolge sortiert.

Die Spalte TIME STAMP enthält den Zeitstempel des entsprechenden Aufrufs, der protokolliert wurde (in Millisekunden).

Die Liste TRACE erfasst folgende Ereignisse und Daten:

- den aufgerufenen Transaktionscode (TAC)
- die Transaktions-ID. In *open*UTM wird für jede Transaktion eine eindeutige Transaktions-ID vergeben. Dieses Kennzeichen wird auch den angeschlossenen Datenbanken an der UTM-DB-Schnittstelle übergeben. Damit wird es u.a. möglich, Traces der Datenbank mit diesen Traces von *open*UTM zu koppeln und Zusammenhänge zwischen UTM- und DB-Abläufen herzustellen. Die Transaktions-ID besteht aus vier Teilen:

  SC Session Counter: er nummeriert die Anwendungsläufe. Nach Neugenerierung ist er 1, bei jedem Start der Anwendung wird er um 1 hochgezählt.

  VC Vorgangs Counter: er nummeriert die Vorgänge innerhalb eines Anwendungslaufes und läuft bis 16 777 216 (2^{24}).

  TC Transaction Counter: er nummeriert die Transaktionen innerhalb eines Vorgangs und läuft bis 32 768 (2^{15}).

  VN Vorgangs Nummer: das ist bei einem Dialogvorgang die bei der UTM-Generierung vergebene Nummer des USERs, bei einem Asynchronvorgang die beim Anwendungsstart vergebene Nummer des UTM-Prozesses.

  Diese vier Teile werden nach dem KDCS-Aufruf INIT protokolliert.
  Für den Anwender sind die Angaben zu VC und TC interessant.

- Alle UTM-Aufrufe mit ihren Operationsmodifikationen. Auch UTM-interne Aufrufe (WAIT, CONT, ...) werden aufgeführt. Sehen Sie auch die KCOP-Liste.

  Zusätzlich wird protokolliert:

  - KCMF bei Aufrufen, bei denen KCMF relevant ist
  - KCRN bei Aufrufen, bei denen KCRN relevant ist
  - KCLT bei den Aufrufen PADM/DADM
  - bei einem Abbruch mit PEND ER/ FR als Diagnoseinformation:
    - der TAC des Teilprogramms, das den Abbruch verursacht hat
    - die Returncodes KCRCDC und KCRRCC
    - VC und TC, für die Zuordnung zum abgebrochenen Vorgang
  - bei einem Aufruf PEND RS als Diagnoseinformation:
    - der TAC des aktuellen Teilprogramms
    - VC und TC, für die Zuordnung zum abgebrochenen Vorgang
  - bei DB-Aufrufen FITA und CATA
    - VC und TC, für die Zuordnung zum abgebrochenen Vorgang

Solange kein Prozesswechsel stattfindet, sind alle Aufrufe zur Bearbeitung eines Dialogschrittes in der gleichen TSN-Spalte hintereinander aufgeführt. Nach PEND PA/PR kann ein Prozesswechsel nur bei einem Wechsel der TAC-Klasse auftreten. Die Unterbrechung eines Prozesses durch das Betriebssystem wird dadurch sichtbar, dass mitten in der Bearbeitung eines Dialogschrittes die Aufrufe in einer anderen Prozess-Spalte fortgesetzt werden.

TRACE2: TASK PERFORMANCE TRACE

Die Auswertungsliste TRACE2 listet die wichtigsten Ereignisse in den Teilprogrammen der Anwendung sequenziell auf. Da die Auswertung nicht wie in der Liste TRACE in Spalten für die UTM-Prozesse erfolgt, kann die Liste TRACE2 beliebig viele Prozesse auswerten. TRACE2 enthält zusätzlich zu den Einträgen der Auswertung TRACE wichtige Daten zur Performanceanalyse.

Die Einträge in der Auswertung sind in zeitlicher Reihenfolge sortiert. Die Spalte TIME STAMP enthält den Zeitstempel des Ereignisses (Genauigkeit: Millisekunden).

Die Auswertungsliste TRACE2 erfasst folgende Ereignisse und Daten:

- Start des Teilprogramms als Eintrag `strt >>>` *tac* mit
  - Transaktionscode des Teilprogramms
  - TAC-Klasse
  - Wartezeit des Auftrags in der TAC-Klasse
- Alle UTM-Funktionsaufrufe mit Operationscode und -modifikation und zusätzlich folgende Informationen:
  - KCMF bei Aufrufen, bei denen KCMF relevant ist
  - KCRN bei Aufrufen, bei denen KCRN relevant ist
  - KCLT bei den Aufrufen PADM und DADM
  - Bei PEND-Aufrufen mit KCOM = ER/FR/RS und bei den Datenbankaufrufen FITA und CATA die Transaction-ID (SC,VC,TC und VN) für die Zuordnung zum abgebrochenen Vorgang.
  - Wenn KCRCCC ≠ 0, die Returncodes KCRCCC und KCRCDC und die Transaction-ID (SC,VC,TC und VN).
- Ende des Teilprogramms als Eintrag `WAIT end<<<<` mit
  - CPU-Verbrauch im Teilprogramm in Millisekunden in Spalte „CPU“

Die Strukturelemente <<<<<< in der Liste erleichtern die Lesbarkeit der Einträge.

12 Anhang

12.1 *open*UTM installieren in UNIX

Bevor Sie UTM-Anwendungen an Ihrem System erzeugen können und UTM-Anwendungen ablaufen können, müssen Sie *open*UTM selbst im System installieren.

Unter „*open*UTM“ versteht man die UTM-Systemfunktionen (Systemcode), C-Includes und COBOL-Copys zum Erzeugen des Anschlussprogramms, Programme für die Dialogterminalprozesse, die Druckprozesse, die Timerprozesse (Zeitsteuerung) die Netzprozesse sowie die Tools zum Erstellen, Betreiben und Verändern von UTM-Anwendungen.

i Wenn Sie mehrere Festplatten an Ihrem Rechner betreiben, sollten Sie aus Performancegründen die UTM-Anwendung und das Datenbanksystem auf verschiedene Platten legen.

Die Programme für die Dialogterminalprozesse (Dialog-Terminal-Programm) und für die Druckerprozesse (Printer-Programm) werden bei der Installation automatisch gebunden. Das Binden erfolgt mit Hilfe von Shellskripts, die zum Lieferumfang von *open*UTM gehören. Das Dialogterminalprogramm und das Printer-Programm können Sie mit Hilfe dieser Shellskripts auch nachträglich binden.

Bei der Installation wird außerdem der COBOL- bzw. C++-Anschlussmodul übersetzt. Zum nachträglichen Übersetzen gibt es auch Shellskripts.

In den folgenden Abschnitten ist beschrieben, wann diese Programme neu zu binden sind und was Sie beim Binden beachten müssen.

12.1.1 UTM-Systemfunktionen in UNIX installieren

Wie Sie *open*UTM in Ihrem System installieren müssen, ist abhängig vom Betriebssystem. Näheres zur Installation finden Sie in der Freigabemitteilung und der Lieferinformation. In der Freigabemitteilung sind auch die Versionsabhängigkeiten zu Produkten aufgeführt, mit denen *open*UTM zusammenarbeitet. In der Lieferinformation stehen Kommandos zum Installieren.

UTM-Dateiverzeichnis

Das Verzeichnis mit den für den Ablauf von *open*UTM notwendigen Dateien wird in diesem Handbuch als **utm-dateiverzeichnis** bezeichnet.

Für den Pfadnamen von *utm-dateiverzeichnis* gilt:

- Wird *open*UTM auf einer Plattform nur im 32-bit-Modus ausgeliefert, dann wird bei der Installation folgendes Verzeichnis eingerichtet:

  utm-installationsverzeichnis`/utm52a00` für den 32-bit-Modus

- Wird *open*UTM auf einer Plattform sowohl im 32-bit- als auch im 64-bit-Modus ausgeliefert, dann werden bei der Installation zwei Verzeichnisbäume eingerichtet:

  utm-installationsverzeichnis`/utm52a00/32` für den 32-bit-Modus

  utm-installationsverzeichnis`/utm52a00/64` für den 64-bit-Modus

utm-installationsverzeichnis ist das Verzeichnis, das bei der Installation angegeben wurde. `utm52a00` bezeichnet die aktuelle Version. Diese kann sich mit einer neuen Korrekturstufe ändern. Beachten Sie daher die Freigabemitteilung.

Um einen korrekten Ablauf von *open*UTM zu garantieren, müssen Sie die Umgebungsvariable UTMPATH auf den Wert von *utm-dateiverzeichnis* setzen, siehe auch Abschnitt „Starten einer UTM-Anwendung in UNIX“ auf Seite 53.

Beispiel

– *open*UTM wird unter `/opt/lib` auf einem Solaris-System für den Ablauf im 32-bit- und 64-bit-Modus installiert. *utm-dateiverzeichnis* ist damit `/opt/lib/utm52a00/32` (32-bit-Modus) bzw. `/opt/lib/utm52a00/64` (64-bit-Modus).

– *open*UTM wird auf einer 32-bit-Plattform unter `/opt/lib` installiert. *utm-dateiverzeichnis* ist damit `/opt/lib/utm52a00`.

12.1.2 Dialogterminal- und Printer-Programm binden

12.1.2.1 Standard-Binderlauf bei Installation

Bei der UTM-Installation werden das Dialogterminalprogramm *utmdtp* und das Printer-Programm *utmprint* automatisch gebunden.

Bei der UTM-Installation werden zum Binden folgende Shellskripts aufgerufen, die mit dem Produkt ausgeliefert und im Dateiverzeichnis *utm-dateiverzeichnis*`/shsc` abgelegt werden:

- `lnkdtp.NONE`
  Binderlauf für das Dialogterminalprogramm *utmdtp* ohne Formatierung
- `lnkprint.NONE`
  Binderlauf für das Printerprogramm *utmprint* ohne Formatierung

12.1.2.2 Programme ohne Formatierung nachträglich binden

Hierzu sind folgende Schritte notwendig:

1. Anmelden unter der Systemverwalterkennung (login unter root)
2. `cd` *utm-dateiverzeichnis*
3. *utmdtp* bzw. *utmprint* löschen mit `rm ./ex/utmdtp` bzw. `rm ./ex/utmprint`
4. `./shsc/lnkdtp.NONE` bzw. `./shsc/lnkprint.NONE` aufrufen
5. Weitere Angaben (z.B. Ziel-Dateiverzeichnis) werden vom jeweiligen Shellskript interaktiv angefordert

Treten während des Bindens Fehler auf, dann stehen im Ziel-Dateiverzeichnis die *stdout*- und *stderr*-Dateien `dtp.ld`[1-2]`.e` bzw. `prt.ld`[1-2]`.e` zur Fehleranalyse zur Verfügung.

12.1.3 Unterschiedliche Socket-Netzprozesse einsetzen

Bei der Installation von *open*UTM werden in *utm-dateiverzeichnis*`/ex` mehrere Socket-Netzprozesse bereitgestellt. Sie unterscheiden sich im wesentlichen durch die maximale Anzahl der Socket-Verbindungen, die parallel aktiv sein können. Die jeweilige maximale Anzahl ist im Dateinamen enthalten, z.B. ist *utmnets1024* der Socket-Netzprozess für bis zu 1024 Socket-Verbindungen.

Nach der Installation ist standardmäßig der Prozess für bis zu 1024 parallele Verbindungen im Einsatz. Der aktuell aktive Socket-Netzprozess hat immer den Namen *utmnets*.

Wenn Sie einen anderen Socket-Netzprozess einsetzen möchten, z.B. *utmnets2000* für bis zu 2000 Socket-Verbindungen, dann gehen Sie wie folgt vor:

1. Beenden Sie die UTM-Anwendung
2. Kopieren Sie *utm-dateiverzeichnis*`/ex/utmnets2000` nach *utm-dateiverzeichnis*`/ex/utmnets`
3. Starten Sie die UTM-Anwendung wieder

12.2 *open*UTM installieren unter Windows

Bevor Sie UTM-Anwendungen an Ihrem System erzeugen können und UTM-Anwendungen ablaufen können, müssen Sie *open*UTM selbst im System installieren.

Unter „*open*UTM“ versteht man die UTM-Systemfunktionen (Systemcode), C-Includes zum Erzeugen des Anschlussprogramms, Programme für die Dialogterminalprozesse, die Timerprozesse (Zeitsteuerung), die Netzprozesse sowie die Tools zum Erstellen, Betreiben und Verändern von UTM-Anwendungen.

Die Hardware- und Software-Voraussetzungen entnehmen Sie bitte der Freigabemitteilung.

12.2.1 Installation des *open*UTM-Servers V5.2

Die Installation der UTM-Systemfunktionen kann nur unter einer Kennung mit Administrator-Rechten vorgenommen werden. Es sind folgende Schritte notwendig:

1. Stellen Sie sicher, dass Microsoft Visual C++ installiert ist.
2. Starten Sie das Programm `SETUP.EXE` auf der *open*UTM-Installations-CD-ROM.
3. Wählen Sie die zu installierenden Produkte aus und installieren Sie - wenn nicht schon vorhanden - PCMX-32.

   PCMX-32 wird auch von anderen Produkten verwendet und kann deshalb bereits in einer anderen Korrekturstufe an Ihrem Rechner vorhanden und für bestimmte Netzverbindungen vorkonfiguriert sein. Es wird jedoch empfohlen, die neueste Version von PCMX-32 zu verwenden.
4. Folgen Sie den weiteren Anweisungen des Installationsprogramms und wählen Sie die passenden Optionen.

   *open*UTM überprüft die Systemvoraussetzungen und die Verfügbarkeit von ausreichender Plattenspeicherkapazität. Falls die Überprüfung negativ ausfällt, wird die Installation abgelehnt.

   Falls eine ältere Version von *open*UTM auf dem PC vorhanden ist, wird diese Version automatisch aktuallisiert.
5. Entnehmen Sie nach der Installation die CD aus dem Laufwerk und booten sie das Systen neu.

*open*UTM wählt das Installations-Dateiverzeichnis *utm-dateiverzeichnis* abhängig von der Eingabe während der Installation.

12.2.2 Benutzerumgebung

Für das Entwickeln von Anwendungen sind keine Administratorrechte erforderlich.

Einige Tools (z.B. KDCDEF) müssen in einem DOS-Eingabeaufforderungs-Fenster aufgerufen werden. Um diese trotzdem per Mausklick starten zu können, können Shortcuts eingerichtet werden Auf Seite 56 finden Sie ein Beispiel für das Starten einer UTM-Anwendung über Shortcut; weitere Shortcut-Beispiele siehe Quick Start Kit. Mehr Informationen zu Shortcuts finden Sie in der Windows-Dokumentation.

12.3 Struktur des UTM-Installationsverzeichnisses

*open*UTM legt bei der Installation folgende Dateien im *utm-dateiverzeichnis* an:

Datei	Beschreibung
applifile	Allgemeine Kontrolldatei für alle UTM-Anwendungen
applifile.bak	Kopie der APPLIFILE (nur bei Windows)
msgdescription	Meldungsdefinitionsdatei
mtxtin	Eingabedatei für das Programm KDCMTXT
install.log	Protokolldatei der Installation bei Windows
utm.log	Protokolldatei der Installation bei UNIX
unwise.exe	Deinstallationsprogramm (nur bei Windows)

*open*UTM legt bei der Installation folgende Verzeichnisse im *utm-dateiverzeichnis* an:

Verzeichnis	Beschreibung
docs	Dokumentation
copy-cobol85	Copy Elemente für COBOL
cpic	CPI-C Schnittstelle
diaglst	Datenstrukturen für die Diagnose
ex	Tools und Programme
excel	Excel-Makros
format	Formate von *open*UTM (nur bei UNIX)
icons	Icons (nur bei Windows)
include	Header-Files
log	Protokoll-Dateien (nur bei Windows)
nls	National Language Support, Meldungskataloge
oss	Produkt OSS (nur bei UNIX)
sample	*open*UTM Beispieldateien
shsc	Verzeichnis mit UTM-Shellskripts und -Prozeduren
src	Quellcode für Änderungen bzw. Debugging
sys	Objektbibliotheken und Objektmodule
tx	Produkt TX
upicl	lokales UPIC-Verzeichnis
xatmi	XATMI für Clients mit Trägersystem UPIC

12.4 Umgebungsvariablen einer UTM-Anwendung

In diesem Abschnitt werden alle Umgebungsvariablen aufgelistet, die für die Steuerung von *open*UTM von Bedeutung sind. Sie sind aufgeteilt in folgende Gruppen:

- allgemeine Umgebungsvariablen, die von UTM-Tools (z.B. KDCDEF, KDCUPD) und beim Start der UTM-Prozesse ausgewertet werden.
- Umgebungsvariablen, die im Betrieb der Anwendung innerhalb der Workprozesse ausgewertet werden.
- Umgebungsvariablen, die von den Netzprozessen einer UTM-Anwendung ausgewertet werden.
- Umgebungsvariable für das Tool KDCDUMP.
- Umgebungsvariablen für die X/Open-Schnittstellen CPIC, XATMI und TX.
- Zusätzliche Umgebungsvariablen für *open*UTM auf Windows.

Zu jeder Umgebungsvariable werden Bedeutung, Wertebereich, Standardeinstellung und der Prozess bzw. das UTM-Modul angegeben, der/das die Umgebungsvariable auswertet.

Alle Umgebungsvariablen müssen vor dem Start der UTM-Anwendung gesetzt werden.

12.4.1 Allgemeine Umgebungsvariablen für *open*UTM unter UNIX und Windows

UTMPATH

Bedeutung

Dateiverzeichnis, in dem die ausführbaren Programme von *open*UTM sowie die Datei `applifile` stehen. Dies erleichtert den Zugriff auf *open*UTM Dateien und Verzeichnisse.

Wertebereich

Dateiverzeichnis, unter dem die für den Ablauf von *open*UTM notwendigen Dateien stehen (*utm-dateiverzeichnis*, siehe Seite 186).

Standardwert
UNIX: Kein Standardwert, UTMPATH muss immer gesetzt werden.
Windows: UTMPATH wird bei der Installation gesetzt.

Modul/Prozess
Wird von jedem UTM-Prozess und beim Start einiger UTM-Tools ausgewertet.

UTMTRAC

Bedeutung
Schaltet optional den dynamischen Trace ein.

Wertebereich
Auswahl der zu protokollierenden UTM-Programme und der Trace-Units. Syntax siehe Handbuch „Meldungen, Test und Diagnose“.

Standardwert
Keiner. Falls nicht gesetzt, wird kein dynamischer Trace erzeugt.

Modul/Prozess
Wird in jedem UTM-Prozess beim Starten des Prozesses ausgewertet.

LANG

Bedeutung
Sprache, in der die UTM-Meldungen ausgegeben werden.

Wertebereich
Sprachkennzeichen z.B. De_DE.646. Für die Sprache muss ein NLS-Katalog vorhanden sein.

Standardwert
Wird LANG nicht oder falsch gesetzt (z.B. zum angegebenen Sprachkennzeichen wird keine NLS- Kataloge gefunden), dann werden die Meldungen in englisch ausgegeben.

Modul/Prozess
Wird in jedem UTM-Prozess, der UTM-Meldungen ausgibt, beim Starten des Prozesses ausgewertet.

UTM_IPC_LETTER

Bedeutung
Legt die Größe des Datenbereichs im IPC-Shared-Memory fest. Der Datenbereich dient zur Ablage der Nachrichten, die zwischen den Prozessen einer Anwendung ausgetauscht werden. In UTM_IPC_LETTER geben Sie die Anzahl der 4KB-Einheiten an, die der Datenbereich umfassen soll.

Wertebereich
Minimum: 5 (d.h. 20KB)

Standardwert
Abhängig von der Anzahl der generierten Semaphore.

Modul/Prozess
Wird im ersten Workprozess beim Starten einer UTM-Anwendung ausgewertet.

UTM_CORE_DUMP

Bedeutung

Verhindert, dass in UNIX Core-Dumps erzeugt werden,

– wenn es im Workprozess zu einem UTM-Dump kommt
– oder wenn externe UTM-Prozesse abnormal beendet werden.

Wertebereich

Falls UTM_CORE_DUMP mit "NO" belegt, wird kein core-Dump erzeugt.

Standardwert

Keiner. Falls nicht gesetzt oder nicht mit "NO" belegt, wird in den oben beschriebenen Situationen ein Core-Dump erzeugt.

Modul/Prozess

Wird in jedem Prozess einer UTM-Anwendung beim Anfordern eines core-Dumps ausgewertet.

UTM_MSG_DATE

Bedeutung

Verhindert, dass den UTM-Meldungen mit Meldungsziel STDOUT und STDERR das Datum und die Uhrzeit vorangestellt werden.

Wertebereich

Falls die Umgebungsvariable gesetzt ist und den Wert "NO" enthält, werden bei Meldungsausgabe nach STDOUT und STDERR weder Datum noch Uhrzeit vorangestellt.

Standardwert

Keiner. Falls die Umgebungsvariable nicht gesetzt ist oder nicht den Wert "NO" enthält, werden allen UTM-Meldungen Datum und Uhrzeit zur besseren Diagnose vorangestellt. Eine Ausnahme sind Meldungen von UTM-Tools. Bei diesen werden nie das Datum und Uhrzeit vorangestellt.

Modul/Prozess

UTM_MSG_DATE wird in jedem Prozess beim Starten des Prozesses ausgewertet.

UTM_IPC_EXTP_LETTER

Bedeutung

Legt die Größe des Datenbereichs im IPC-Shared-Memory fest, der pro Verbindung maximal zur Verfügung steht. Der in UTM_IPC_EXTP_LETTER angegebene Wert wird als Anzahl von 4KB-Bereichen interpretiert. Siehe auch Seite 50.

Wertebereich

Minimum: 1 (d.h. 4KB)

Standardwert

16 (d.h. die 64KB).

Modul/Prozess

Wird im ersten Workprozess beim Starten der Anwendung ausgewertet.

12.4.2 Umgebungsvariablen für Workprozesse

KDCS_C_DEBUG

Bedeutung
Ist KDCS_C_DEBUG gesetzt, dann wird jeder Aufrufe eines C/C++- oder COBOL-Teilprogramms und jeder KDCS-Aufrufen in einem C/C++-Teilprogramm auf *stdout* protokolliert.

Wertebereich
Falls gesetzt, wird die Protokollierung aktiviert.

Standardwert
Keine Protokollierung.

Modul/Prozess
Wird in jedem Workprozess einer UTM-Anwendung beim ersten Aufruf eines C/C++- oder COBOL-Teilprogramms ausgewertet.

UTM_ABORT_WITH_EXCEPTION

Bedeutung
Legt fest, ob bei einem Anwendungsabbruch auf UNIX ein core erzeugt bzw. auf Windows der Debugger aktiviert wird. UTM_ABORT_WITH_EXCEPTION sollte nur zusammen mit dem Startparameter STXIT=OFF verwendet werden.

Wertebereich
Falls gesetzt, wird bei Anwendungsabruch ein core erzeugt bzw. der Debugger aktiviert.

Standardwert
Kein Erzeugen eines cores bzw. kein Aktivieren des Debuggers.

Modul/Prozess
Wird in jedem Workprozess beim Anwendungsabbruch ausgewertet.

PATH

Bedeutung
Legt die Pfade fest, unter denen die Shellskripts *admlp* und *utmlp* gesucht werden. Das *admlp*-Skript soll beim Drucken von EAM-Dateien verwendet werden. Das *utmlp*-Skript wird beim Drucken im Printerprozess *utmprint* verwendet.

Wertebereich
Anzahl von Dateiverzeichnissen (in der Form *datei-verzeichnis1*:*datei-verzeichnis2*:...)

Standardwert
Wird unter den in PATH angegebenen Dateiverzeichnissen das *admlp*- bzw. *utmlp*-Skript nicht gefunden, dann wird das *admlp*- bzw. *utmlp*-Skript unter $UTMPATH`/shsc` verwendet.

Modul/Prozess
Wird in jedem Workprozess beim Drucken von EAM-Dateien und in jedem Druckerprozess beim Starten des Prozesses ausgewertet.

UTM_OSS_SHM_BASE

Bedeutung
Legt optional die Basisadresse des OSS-Shared-Memory im Workprozess fest. Hierdurch können Adresskonflikte beim Einrichten dieses Datenbereiches vermieden werden. Dies ist z.B. die Ursache des Fehlers 91 beim Anwendungsstart.

Wertebereich
Gültige Adresse, an die das OSS-Shared-Memory im Prozess geladen werden kann.
Empfohlene Werte für NT: Adresse zwischen 0x30000000 und 0x70000000.
Diese Adresse muss ein 64KB Segment sein, d.h. mit 0x0000 enden.

Standardwert
Wird UTM_OSS_SHM_BASE nicht oder auf den Wert 0x00000000 gesetzt, wird der Bereich auf die nächste verfügbare Adresse gelegt.

Modul/Prozess
Die Umgebungsvariable wird in jedem Workprozess mit OSI TP beim Starten des Prozesses ausgewertet.

12.4.3 Umgebungsvariablen für Netzprozesse

UTM_APPLI_CONLET

Bedeutung
Mit UTM_APPLI_CONLET können NEA-Protokolle im UTM Netz-Multiprozess unterstützt werden.

Wertebereich
NEA Connection Letter für die Verbindungsaufbauphase.

Standardwert
Falls nicht gesetzt, wird die Funktionalität der CMX-Progammschnittstelle (ICMX(L)) verwendet.

Modul/Prozess
UTM_APPLI_CONLET wird in jedem Netzprozess beim Starten des Prozesses ausgewertet.

UTM_NET_COMMON_WAIT

Bedeutung
Durch UTM_NET_COMMON_WAIT kann der Durchsatz bei den Netzanbindungen erhöht werden.
Diese Optimierung kann nur bei ausschließlicher Verwendung von TCP/IP Netzen eingesetzt werden.

Wertebereich
Die Umgebungsvariable wird gesetzt oder nicht.

Standardwert
Optimierung eingeschaltet (die Umgebungsvariable wird bei der Installation gesetzt).

Modul/Prozess
UTM_NET_COMMON_WAIT wird in jedem Netzprozess beim Starten des Prozesses ausgewertet.

UTM_NET_SELECT_TIME

Bedeutung
Legt die maximale Verweilzeit in Millisekunden im `select()`-Aufruf von PCMX fest. Hierdurch können Netzwerkzugriffe beschleunigt werden.

Wertebereich
Gültige Werte sind 1 - 999(msec).

Standardwert
100(msec).

Modul/Prozess
UTM_NET_SELECT_TIME wird in jedem Netzprozess beim Starten des Prozesses ausgewertet.

12.4.4 Umgebungsvariable für das Tool KDCDUMP

EDITOR

Bedeutung
Enthält den Editor, der vom Tool KDCDUMP beim Kommando `EDT` aufgerufen wird. Siehe *open*UTM-Handbuch „Meldungen, Test und Diagnose (UNIX, Windows)“.

Wertebereich
Editorprogramm; z.B. vi, Pfe, notepad

Standardwert
UNIX: vi
Windows: wordpad

Modul/Prozess
EDITOR wird von KDCDUMP beim Kommando `EDT` ausgewertet.

12.4.5 Umgebungsvariablen für die X/Open Schnittstellen

Im Folgenden sind die Umgebungsvariablen aufgelistet, mit denen Sie CPI-C-, TX- und XATMI-Anwendungen steuern können. Eine detaillierte Beschreibung zu diesen Umgebungsvariablen finden Sie im *open*UTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

12.4.5.1 CPIC

Die Umgebungsvariablen CPICTRACE, CPICBUFFER und CPICDUMP steuern den CPIC-Trace. Der Trace wird durch Setzen mindestens einer der Umgebungsvariablen eingeschaltet.
Für die Umgebungsvariablen gilt Folgendes:

Wertebereich
Eine oder mehrere der Umgebungsvariablen werden gesetzt oder nicht. Werden CPICDUMP und/ oder CPICBUFFER gesetzt, dann gilt CPICTRACE auch als gesetzt.

Standardwert
Falls keine der Umgebungsvariablen gesetzt ist, dann wird kein CPI-C-Trace erzeugt.

Modul/Prozess
Die Umgebungsvariablen werden in jedem Workprozess beim Starten des Prozesses ausgewertet.

CPICTRACE

Umfang der Protokollierung

Zu jedem CPI-C-Funktionsaufruf wird der Inhalt der Input- und Output-Parameter ausgegeben. Von den Datenpuffern werden nur die ersten 16 byte ausgegeben. Die Returncodes der KDCS-Aufrufe, auf die die CPI-C-Aufrufe abgebildet werden, werden ausgegeben.

CPICBUFFER

Umfang der Protokollierung

Wie bei CPICTRACE. Die Datenpuffer werden jedoch in voller Länge protokolliert.

CPICDUMP

Umfang der Protokollierung

Wie bei CPICTRACE. Zusätzlich werden Diagnose-Information in die Trace-Datei geschrieben.

Sind beim Start der Anwendung CPICDUMP und CPICBUFFER gesetzt, dann werden sowohl die Datenpuffer in voller Länge als auch die zusätzliche Diagnose-Information in die Trace-Datei geschrieben.

12.4.5.2 TX

Die Umgebungsvariablen TXTRACE und TXPATH steuern den TX-Trace.

TXTRACE

Bedeutung

Schaltet den TX-Trace ein und steuert den Umfang der Diagnoseinformation, die protokolliert wird.

Wertebereich

Folgende Wert können Sie für TXTRACE angeben:

E Nur Fehler werden protokolliert. (Error)

I Zusätzlich zu „E“ werden alle TX-Aufrufe protokolliert. (Interface)

F Zusätzlich zu „I“ werden alle KDCS-Aufrufe, auf die die TX-Aufrufe abgebildet werden, protokolliert. (Full)

D Zusätzlich zu „F“ werden Diagnose-Informationen protokolliert. (Debug)

Standardwert

Falls nicht gesetzt ist, wird kein TX-Trace erzeugt.

Modul/Prozess

TXTRACE wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

TXPATH

Bedeutung
Enthält das Dateiverzeichnis, in das die Trace-Datei des TX-Trace geschrieben wird.

Wertebereich
Dateiverzeichnis

Standardwert
Dateiverzeichnis, in dem die Anwendung gestartet wurde

Modul/Prozess
TXPATH wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

12.4.5.3 XATMI

Für XATMI-Anwendungen werden die im Folgenden aufgelisteten Umgebungsvariablen ausgewertet.

XTLCF

Bedeutung
Enthält den Namen der verwendeten Local Configuration File (LCF)

Wertebereich
Dateiname bzw. Pfadname, der den Konventionen des Betriebssystems entspricht. Wenn XTPALCF verwendet wird, darf XTLCF nur einen Dateinamen enthalten.

Standardwert
`xatmilcf` im Dateiverzeichnis, in dem die Anwendung gestartet wurde.

Modul/Prozess
XTLCF wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

XTPALCF

Bedeutung
Legt die Dateiverzeichnisse fest, unter denen *open*UTM zusätzliche Beschreibungen von typisierten Puffern sucht. Siehe auch *open*UTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Wertebereich
Dateiverzeichnisse, angegeben in folgender Form:
UNIX: *verzeichnis1*:*verzeichnis2*:...(getrennt durch Doppelpunkt)
Windows: *verzeichnis1*;*verzeichnis2*;...(getrennt durch Semikolon)

Standardwert
Es wird nur die LCF in XTLCF durchsucht (bzw. die Datei `xatmilcf`, wenn XTLCF nicht gesetzt ist).

Modul/Prozess
XTPALCF wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

XTPATH

Bedeutung
Enthält das Dateiverzeichnis, in das die Trace-Datei des XATMI-Trace geschrieben wird.

Wertebereich
Dateiverzeichnis

Standardwert
Dateiverzeichnis, in dem die Anwendung gestartet wurde

Modul/Prozess
XTPATH wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

XTSVRTR

Bedeutung
Schaltet den XATMI-Trace ein und steuert den Umfang der Diagnoseinformation, die protokolliert wird.

Wertebereich
Folgende Wert können Sie für XTSVRTR angeben:
E Nur Fehler werden protokolliert. (Error)
I Zusätzlich zu „E“ werden alle XATMI-Aufrufe protokolliert. (Interface)
F Aktiviert den vollen XATMI-Trace (Full)

Standardwert
Falls nicht gesetzt, wird kein XATMI-Trace erzeugt.

Modul/Prozess
XTSVRTR wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

12.4.6 Zusätzliche Umgebungsvariablen für *open*UTM unter Windows

USERNAME

Bedeutung
Enthält die UTM-Benutzerkennung für den Dialog-Terminalprozess *utmdtp*.

Wertebereich
UTM-Benutzerkennung

Standardwert
Windows-NT-Benutzerkennung, unter der der *utmdtp* gestartet wird.

Modul/Prozess
USERNAME wird in jedem *utmdtp* beim Starten des Prozesses ausgewertet.

UTM_PIPE_TIME

Bedeutung
Enthält die Wartezeit in Sekunden bzgl. des named pipe handlings auf Windows.

Wertebereich
1 - 99 (sec).

Standardwert
3 (sec).

Modul/Prozess
UTM_PIPE_TIME wird in jedem UTM-Prozess, der named pipes verwendet, beim Starten des Prozesses ausgewertet.

UTM_MAIN_KILL_TIME

Bedeutung
Enthält die maximale Zeit in Sekunden, die auf die normale Beendigung von Timerprozess und Netzprozess(en) gewartet wird.

Wertebereich
1 bis 99(sec)

Standardwert
10 (sec)
Sie können das Beenden der UTM-Anwendung beschleunigen, wenn Sie UTM_MAIN_KILL_TIME auf einen kleineren Wert setzen.

Modul/Prozess
UTM_MAIN_KILL_TIME wird in jedem Mainprozess beim Starten des Prozesses ausgewertet.

UTM_BREAK_BEFORE_DUMP

Bedeutung
Legt fest, ob vor jedem UTM-Dump ein Breakpoint angefordert wird.

Wertebereich
Umgebungsvariable definiert oder nicht definiert. Es ist kein Wert notwendig.
Ist die Umgebungsvariable definiert, dann wird vor jedem UTM-Dump ein Breakpoint angefordert. In diesem Fall kann der Anwendungslauf unter der Kontrolle des Microsoft Visual C++ Debuggers fortgesetzt werden, um den Fehler zu analysieren. Dabei wird der UTM-Dump erzeugt (wie ohne Breakpoint).

Standardwert
Umgebungsvariable nicht definiert, d.h. kein Breakpoint.

Modul/Prozess
Die Umgebungsvariable wird vor jedem UTM-Dump ausgewertet.

UTM_BREAK_BEFORE_KCSTRMA

Bedeutung
Legt fest, ob vor jedem Anwendungsabbruch ein Breakpoint angefordert wird.

Wertebereich
Umgebungsvariable definiert oder nicht definiert. Es ist kein Wert notwendig.
Ist die Umgebungsvariable definiert, dann wird vor jedem Anwendungsabbruch ein Breakpoint angefordert. In diesem Fall kann der Anwendungslauf unter der Kontrolle des Microsoft Visual C++ Debuggers fortgesetzt werden, um den Fehler zu analysieren. Dabei wird der UTM-Dump erzeugt (wie ohne Breakpoint).

Standardwert
Umgebungsvariable nicht definiert, d.h. kein Breakpoint.

Modul/Prozess
Die Umgebungsvariable wird vor jedem Anwendungsabbruch ausgewertet.

TEMP, **TMP**

Bedeutung
enthält das Dateiverzeichnis, in das *open*UTM beim TNS-Check eine temporäre Datei schreiben kann.

Wertebereich
Dateiverzeichnis

Standardwert
`\tmp`

Modul/Prozess
TEMP bzw. TMP wird im UTM-Programm, das die TNS-Einträge auf Windows prüft, beim Starten des Programms ausgewertet.

12.5 Aufbau der Accountingsätze von *open*UTM

Die Accountingsätze von *open*UTM werden in die Abrechnungsdateien im Datenverzeichnis ACCNT geschrieben.

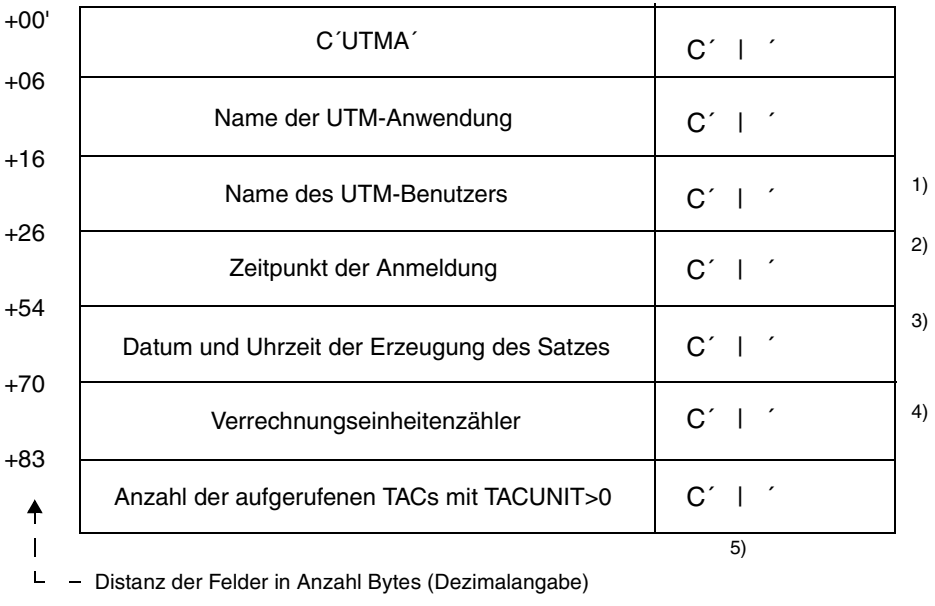
Es gibt folgende zwei Satztypen:

- Abrechnungssätze (Satztyp UTMA)
- Kalkulationssätze (Satztyp UTMK)

In diesem Abschnitt sind die Datenfelder der Sätze beschrieben, die UTM-spezifische Informationen enthalten.

Die Bedeutung dieser Sätze ist im Kapitel „Accounting“ auf Seite 151 beschrieben.

12.5.1 Aufbau des Abrechnungssatzes



Anmerkungen

1) Name des Benutzers. In einer UTM-Anwendung ohne generierte Benutzer trägt *open*UTM den Namen des LTERM-Partners ein.

2) Zeitpunkt der Anmeldung dieses Benutzers (USER) an diesem LTERM in in der Form:
`FRI SEPT 15 00:00:00 2000`
Wenn im aktuellen Lauf der UTM-Anwendung für diesen USER nur Asynchron-TACs aufgerufen wurden, ist der Inhalt dieses Feldes ´------´.

3) Format: *yyyymmddhhmmss* (Jahr/Monat/Tag/Stunde/Minute/Sekunde)

4) Summe der Verrechnungseinheiten für diesen Benutzer, seit der letzte Verrechnungssatz geschrieben wurde bzw. seit dem Zeitpunkt der Anmeldung.

5) Die einzelnen Felder sind durch „ | “ voneinander getrennt.

12.5.2 Aufbau des Kalkulationssatzes

+00	C´ UTMK´	C´ | ´	
+06	Anwendungsname der UTM-Anwendung	C´ | ´	
+16	Transaktionscode des Teilprogramms (TAC)	C´ | ´	
+26	CPU-Zeit in *open*UTM (msec)	C´ | ´	
+36	- reserviert -	C´ | ´	1)
+46	Länge der Eingabenachricht	C´ | ´	
+53	Länge der Ausgabenachricht	C´ | ´	
+60	Anzahl der asynchronen Ausgaben	C´ | ´	
+70	Verrechnungseinheiten für LTAC' s	C´ | ´	2)
+80	Name des UTM-Benutzers	C´ | ´	
+88	Name des LTERM-Partners	C´ | ´	
+96	Realzeit des Teilprogrammlaufs (in msec)	C´ | ´	

↑ – Distanz der Felder in Anzahl Bytes (Dezimalangabe) 3)

Anmerkungen

1) Es wird immer 0 eingetragen.

2) siehe KDCDEF-Anweisung LTAC...,LTACUNIT=

3) Die einzelnen Felder sind durch „ | “ voneinander getrennt.

Beispiel: Abrechnungsphase mit Verteilter Verarbeitung über LU6.1 mit LTACs

Die Anwendung VTV10S im Rechner FILIALE kommuniziert mit Anwendung VTV10S im Rechner ZENTRALE, indem sie LTAC-Aufrufe an die Anwendung VTV10S in ZENTRALE sendet.

Rechner FILIALE

```
UTMK |VTV10S   |CVARL1    |       0 |          |   24 |   73 |       0 |       0 |apu       |BRCV0004 |      50 |
UTMK |VTV10S   |CVARL1    |       0 |          |   51 |   73 |       0 |       0 |apu       |BRCV0004 |      50 |
UTMK |VTV10S   |CVARL1    |       0 |          |    6 |  722 |       0 |       0 |apu       |BRCV0004 |      10 |
UTMK |VTV10S   |CVAR1     |      10 |          |  722 | 2205 |       0 |      50 |apu       |BRCV0004 |     100 |
UTMK |VTV10S   |CVAR2     |      20 |          | 2209 |   89 |       0 |       0 |apu       |BRCV0004 |      40 |
UTMA |VTV10S   |apu       |Fri Sep 15 11:34:50 2000  | 20000915113453 |       1133 |    5 |
UTMK |VTV10S   |KDCBADTC  |      20 |          |    0 |  240 |       0 |       0 |apu       |BRCV0004 |      80 |
UTMA |VTV10S   |apu       |Fri Sep 15 11:34:50 2000  | 20000915113453 |        655 |    0 |
```

Rechner ZENTRALE

Diese Anwendung empfängt die von der Anwendung VTV10S an der FILIALE gesendeten LTACs. Die hier eingetragenen USER- und LTERM-Namen entsprechen dem Session- bzw. dem LPAP-Namen.

```
UTMK |VTV10S   |PERE      |      40 |          |   10 |   80 |       0 |       0 |VTV11S   |LPAP1    |     490 |
UTMK |VTV10S   |RT1       |      30 |          |   10 |   12 |       0 |       0 |VTV12S   |LPAP1    |     200 |
UTMK |VTV10S   |MPF       |      20 |          |   10 |   80 |       0 |       0 |VTV12S   |LPAP1    |     240 |
UTMK |VTV10S   |MPF       |      40 |          |   10 |   80 |       0 |       0 |VTV12S   |LPAP1    |     250 |
UTMK |VTV10S   |RT1       |      30 |          | 2205 | 2209 |       0 |       0 |VTV12S   |LPAP1    |     300 |
UTMK |VTV10S   |RT1       |      50 |          | 4400 | 4404 |       0 |       0 |VTV12S   |LPAP1    |     320 |
UTMK |VTV10S   |RT1       |      30 |          |   10 |   14 |       0 |       0 |VTV12S   |LPAP1    |     220 |
UTMK |VTV10S   |ATAC      |       0 |          |    5 |    0 |       0 |       0 |VTV12S   |LPAP1    |     170 |
...
...
UTMK |VTV10S   |VPASS     |    7940 |          |    8 |    4 |       0 |       0 |ap       |LTP00001 |   84460 |
```

12.6 Druckausgaben ohne Druckersteuerung abwickeln (UNIX)

Die Druckausgabe bei einer *open*UTM(UNIX)-Anwendung wird im Automatikmodus über das Drucker-Shellskript *utmlp* gesteuert. Ist ein Drucker mit einer Anwendung verbunden, so existiert für ihn ein eigener Printerprozess. Erhält ein Drucker und damit ein Printerprozess einen Druckauftrag, dann startet der Printerprozess das Drucker-Shellskript *utmlp*. Die Daten werden in einer Pipe übergeben.

utmlp können Sie bei Bedarf selbst erstellen. Der Printerprozess wertet die Shell-Variable PATH aus und sucht darüber *utmlp*. Findet der Printerprozess darüber kein *utmlp*, dann startet er das Druckerskript *utm-dateiverzeichnis*`/shsc/utmlp`, das mit *open*UTM ausgeliefert wird. Sie können dieses Shellskript anwendungsspezifisch abändern.

utmlp wird vom Printerprozess mit folgenden Argumenten aufgerufen:

$0: vollqualifizierter Name des Drucker-Shellskripts.
Er wird vom Printerprozess mit *pathfind* ermittelt.

$1: *open*UTM(UNIX) Version

$3: *lpr* bzw. *lp*
Abhängig von der Konfigurierung eines Druckers wird entweder der UNIX-Spool (*lpr*) oder der X/Open-Spool (*lp*) verwendet.

$4: Name des Druckers oder der Druckergruppe
(siehe KDCDEF-Steueranweisung PTERM *ptermname* (PRINTER,*class*) im *open*UTM-Handbuch „Anwendungen generieren“).

Tritt bei der Verarbeitung der Daten in *utmlp* ein Fehler auf, dann beendet sich das Druckerskript mit einem Exit-Code ungleich Null. Der Printerprozess erzeugt eine negative Abdruckquittung. Daraufhin baut *open*UTM die Verbindung zu diesem Printerprozess ab und erzeugt die Meldung K046. Der Printerprozess beendet sich.

Alle Ausgabe-Aufträge für einen Drucker werden von *open*UTM in der Message Queue des zugehörigen LTERM-Partners zwischengespeichert. Die zu druckenden Nachrichten gehen also im Fall einer negativen Abdruckquittung nicht verloren. Sie werden beim nächsten Verbindungsaufbau an diesen Drucker gesendet.

12.7 Beispielprogramme und Beispielanwendungen

Für *open*UTM werden standardmäßig Beispielprogramme, Beispielprozeduren sowie ablauffähige Beispielanwendungen ausgeliefert. Diese Beispiele können Sie als Vorlage für die eigene Anwendungsentwicklung nehmen und entsprechend anpassen.

12.7.1 CPI-C-Beispielprogramme

CPI-C-Beispielprogramme finden Sie in UNIX in *utm-dateiverzeichnis*`/cpic/sample`.

Name	**Bedeutung**	
../src/kcpsam1.c	C-Source	→ Asynchroner Teil
../src/kcpsam2.c	C-Source	→ Synchroner Teil
../sys/libcpsam.a	Bibliothek mit Objekten von	→ Asynchroner Teil → Synchroner Teil

Auf Windows sind diese Programme im Quick Start Kit enthalten, siehe Seite 211.

12.7.2 Beispielprozeduren in UNIX

Weitere Produktbestandteile von *open*UTM in UNIX sind die unten angeführten Prozeduren, die Sie beispielsweise beim Installieren oder Erstellen der Anwendung benötigen. Sie können die Prozeduren nach Ihren Wünschen modifizieren und erweitern. Die Prozeduren enthalten englische Kommentare.

Im Dateiverzeichnis *utm-dateiverzeichnis*`/shsc` werden folgende Beispielprozeduren ausgeliefert:

Prozedur	**Funktion**
admlp	Administrationskommandoausgaben mit dem `lp`–Kommando ausdrucken
CCmainutm	erzeugt ein Objekt für den C++-Anschluss
dumpstart	dient zur symbolischen Auswertung eines UTM-Dumps
install.sample	kopiert die Dateien der Beispielanwendung in das Dateiverzeichnis *filebase* der Beispielanwendung
lnkdtp.NONE	bindet das Dialog-Terminal-Programm *utmdtp* ohne Formatierungsfunktionen
lnkprint.NONE	bindet das Druckerprogramm *utmprint* ohne Formatierungsfunktionen
nmwork	erzeugt die Symboltabelle für eine UTM-Produktivanwendung

Prozedur	Funktion
ref4utm	erzeugt eine Eingabedatei für den TNSX-Compiler tnsxcom. tnsxcom erzeugt aus dieser Datei Referenzeinträge im TS-Directory, die das Shellskript *tns4utm* benötigt.
stat2dyn	erzeugt aus einer statischen Bibliothek eine Shared Objects-Bibliothek
tns4utm	erzeugt eine Eingabedatei für den TNSX-Compiler *tnsxcom*. *tnsxcom* erzeugt aus dieser Datei die Einträge im TS-Directory, die Ihre UTM-Anwendung für die verteilte Verarbeitung benötigt. (*tns4utm* wird ab *open*UTM V4.0 nicht mehr weiterentwickelt)
tnsrefin	Eingabe für den TNSX-Compiler, damit dieser den Referenzeintrag `.openUTM` erzeugt.
utmlp	zur Verarbeitung von Druckausgaben

12.7.3 *open*UTM(UNIX)-Beispielanwendung

Zum Lieferumfang von *open*UTM gehört eine Beispielanwendung *utmsample*. Die Datei `CPIO.utmsample` der Beispielanwendung wird zusammen mit *open*UTM installiert und befinden sich in *utm-dateiverzeichnis*.

Sie installieren die Beispielanwendung inklusive Beschreibung unter Ihrer Benutzerkennung, in dem Sie die Prozedur *utm-dateiverzeichnis*`/shsc/install.sample` aufrufen. Aus den Dateien der Beispielanwendung können Sie durch Aufruf einer Shell-Prozedur sofort eine einfache UTM-Anwendung erzeugen.

Mit der Beispielanwendung können Sie sich die Generierung und die Inbetriebnahme Ihrer UTM-Anwendung vereinfachen. Erzeugen Sie dazu aus der Beispielanwendung eine Anwendung, die die Komponenten und Schnittstellen verwendet, die Sie für Ihre Anwendung benötigen (Datenbankanschluss INFORMIX oder ORACLE, verteilte Verarbeitung über OSI TP bzw. LU6.1 etc.). Die Input-Datei für das Generierungstool KDCDEF und die Makefile dieser Anwendung können Sie dann als Vorlage für Ihre Anwendung verwenden.

Zu allen Programmen der Beispielanwendung werden die Quellcodes ausgeliefert, um Ihnen die Programmierung eigener UTM-Programme zu erleichtern.

12.7.4 *open*UTM(Windows) - Quick Start Kit

Mit *open*UTM auf Windows werden UTM-Beispielanwendungen und -Prozeduren in Form eines Quick Start Kit ausgeliefert. Das Quick Start Kit setzt *open*UTM voraus. Bei der Installation von *open*UTM können Sie auswählen, ob das Quick Start Kit gleich mit installiert wird. Sie können es aber auch nachträglich explizit installieren.

Das Quick Start Kit enthält ablauffähige Beispielprogramme, angefangen von einem CPI-C-Client bis zur UTM-Datenbank-Anwendung. Für diese Programme werden bei der Installation Icons eingerichtet, so dass Sie die Programme einfach per Mausklick starten können. Damit lernen Sie die Funktionalität von *open*UTM auf einfache Art und Weise kennen.

Im Quick Start Kit sind auch die zugehörigen Programm-Sourcen, Shellskripts und Konfigurationsdateien enthalten. Diese Sourcen sollen Ihnen die Programmierung eigener Anwendungen erleichtern und können z.B. als Gerüst für eigene Anwendungen hergenommen werden.

Das Quick Start Kit enthält auch eine Dokumentation. Darin sind die ausgelieferten Programme und Source-Dateien mit ihrer Funktion kurz beschrieben.

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.

Ablaufinvariantes Programm
reentrant program
siehe *reentrant-fähiges Programm*.

Abnormale Beendigung einer UTM-Anwendung
abnormal termination of a UTM application
Beendigung einer *UTM-Anwendung*, bei der die *KDCFILE* nicht mehr aktualisiert wird. Eine abnormale Beendigung wird ausgelöst durch einen schwerwiegenden Fehler, z.B. Rechnerausfall, Fehler in der Systemsoftware. Wird die Anwendung erneut gestartet, führt *open*UTM einen *Warmstart* durch.

abstrakte Syntax (OSI)
abstract syntax
Eine abstrakte Syntax ist die Menge der formal beschriebenen Datentypen, die zwischen Anwendungen über *OSI TP* ausgetauscht werden sollen. Eine abstrakte Syntax ist unabhängig von der eingesetzten Hardware und der jeweiligen Programmiersprache.

Access-List
access list
Eine Access-List definiert die Berechtigung für den Zugriff auf einen bestimmten *Service*, auf eine bestimmte *TAC-Queue* oder auf eine bestimmte *USER-Queue*. Eine Access-List ist als *Keyset* definiert und enthält einen oder mehrere *Keycodes*, die jeweils eine Rolle in der Anwendung repräsentieren. Benutzer oder LTERMs dürfen nur dann auf den Service oder die *TAC-Queue*/*USER-Queue* zugreifen, wenn ihnen die entsprechenden Rollen zugeteilt wurden, d.h. wenn ihr *Keyset* und die Access-List mindestens einen gemeinsamen *Keycode* enthalten.

Access Point (OSI)
siehe *Dienstzugriffspunkt*.

ACID-Eigenschaften
ACID properties

Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Administration
administration

Verwaltung und Steuerung einer *UTM-Anwendung* durch einen *Administrator* oder ein *Administrationsprogramm*.

Administrationskommando
administration command

Kommandos, mit denen der *Administrator* einer *UTM-Anwendung* Administrationsfunktionen für diese Anwendung durchführt. Die Administrationskommandos sind als *Transaktionscodes* realisiert.

Administrationsprogramm
administration program

Teilprogramm, das Aufrufe der *Programmschnittstelle für die Administration* enthält. Dies kann das Standard-Administrationsprogramm *KDCADM* sein, das mit *open*UTM ausgeliefert wird, oder ein vom Anwender selbst erstelltes Programm.

Administrator
administrator

Benutzer mit Administrationsberechtigung.

AES

AES (Advanced Encryption Standard) ist der aktuelle symmetrische Verschlüsselungsstandard, festgelegt vom NIST (National Institute of Standards and Technology), basierend auf dem an der Universität Leuven (B) entwickelten Rijndael-Algorithmus. Wird das AES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen AES-Schlüssel.

Akzeptor (CPI-C)
acceptor

Die Kommunikationspartner einer *Conversation* werden *Initiator* und Akzeptor genannt. Der Akzeptor nimmt die vom Initiator eingeleitete Conversation mit Accept_Conversation entgegen.

Anmeldevorgang (KDCS)
sign-on service

Spezieller *Dialog-Vorgang*, bei dem die Anmeldung eines Clients an eine UTM-Anwendung durch *Teilprogramme* gesteuert wird.

Anschlussprogramm
linkage program
siehe *KDCROOT*.

Anwendungsinformation
application information
Sie stellt die Gesamtmenge der von der *UTM-Anwendung* benutzten Daten dar. Dabei handelt es sich um Speicherbereiche und Nachrichten der UTM-Anwendung, einschließlich der aktuell auf dem Bildschirm angezeigten Daten. Arbeitet die UTM-Anwendung koordiniert mit einem Datenbanksystem, so gehören die in der Datenbank gespeicherten Daten ebenfalls zur Anwendungsinformation.

Anwendungsprogramm
application program
Ein Anwendungsprogramm bildet den Hauptbestandteil einer *UTM-Anwendung*. Es besteht aus der Main Routine *KDCROOT* und den *Teilprogrammen*. Es bearbeitet alle Aufträge, die an eine *UTM-Anwendung* gerichtet werden.

Anwendungswiederanlauf
application restart
Start einer *UTM-S*-Anwendung nach einer vorhergehenden abnormalen Beendigung. Dabei wird die *Anwendungsinformation* auf den zuletzt erreichten konsistenten Zustand gesetzt. Unterbrochene *Dialog-Vorgänge* werden dabei auf den zuletzt erreichten *Sicherungspunkt* zurückgesetzt, so dass die Verarbeitung an dieser Stelle wieder konsistent aufgenommen werden kann (*Vorgangswiederanlauf*). Unterbrochene *Asynchron-Vorgänge* werden zurückgesetzt und neu gestartet oder beim zuletzt erreichten *Sicherungspunkt* fortgesetzt. Bei UTM-F-Anwendungen werden beim Start nach einer vorhergehenden abnormalen Beendigung lediglich die dynamisch geänderten Konfigurationsdaten auf den zuletzt erreichten konsistenten Zustand gesetzt.

Application Context (OSI)
application context
Der Application Context ist die Menge der Regeln, die für die Kommunikation zwischen zwei Anwendungen gelten sollen. Dazu gehören z.B. die abstrakten Syntaxen und die zugeordneten *Transfer-Syntaxen*.

Application Entity (OSI)
application entity
Eine Application Entity (AE) repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen global (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title* (AET). Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Title (OSI)
application entity title

Ein Application Entity Title ist ein global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title* des jeweiligen *Application Process* und dem *Application Entity Qualifier*.

Application Entity Qualifier (OSI)
application entity qualifier

Bestandteil des *Application Entity Titles*. Der Application Entity Qualifier identifiziert einen *Dienstzugriffspunkt* innerhalb der Anwendung. Ein Application Entity Qualifier kann unterschiedlich aufgebaut sein. *open*UTM unterstützt den Typ „Zahl“.

Application Process (OSI)
application process

Der Application Process repräsentiert im *OSI-Referenzmodell* eine Anwendung. Er wird durch den *Application Process Title* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (OSI)
application process title

Gemäß der OSI-Norm dient der Application Process Title (APT) zur global (d.h. weltweit) eindeutigen Identifizierung von Anwendungen. Er kann unterschiedlich aufgebaut sein. *open*UTM unterstützt den Typ *Object Identifier*.

Application Service Element (OSI)
application service element

Ein Application Service Element (ASE) repräsentiert eine Funktionsgruppe der Anwendungsschicht (Schicht 7) des *OSI-Referenzmodells*.

Association (OSI)
association

Eine Association ist eine Kommunikationsbeziehung zwischen zwei *Application Entities*. Dem Begriff Association entspricht der *LU6.1*-Begriff *Session*.

Asynchron-Auftrag
queued job

Auftrag, der vom Auftraggeber zeitlich entkoppelt durchgeführt wird. Zur Bearbeitung von Asynchron-Aufträgen sind in *open*UTM *Message Queuing* Funktionen integriert, vgl. *UTM-gesteuerte Queue* und *Service-gesteuerte Queue*. Ein Asynchron-Auftrag wird durch die *Asynchron-Nachricht*, den Empfänger und ggf. den gewünschten Ausführungszeitpunkt beschrieben.
Ist der Empfänger ein Terminal oder Drucker, so ist der Asynchronauftrag ein *Ausgabeauftrag*; ist der Empfänger ein Asynchron-Vorgang derselben oder einer

fernen Anwendung, so handelt es sich um einen *Hintergrundauftrag*. Asynchron-Aufträge können *zeitgesteuerte Aufträge* sein oder auch in einen *Auftragskomplex* integriert sein.

Asynchron-Conversation
asynchronous conversation

CPI-C-Conversation, bei der nur der *Initiator* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein asynchroner Transaktionscode generiert sein.

Asynchron-Nachricht
queued message

Asynchron-Nachrichten sind Nachrichten, die an eine *Message Queue* gerichtet sind. Sie werden von der lokalen *UTM-Anwendung* zunächst zwischengespeichert und dann unabhängig vom Auftraggeber weiter verarbeitet. Je nach Empfänger unterscheidet man folgende Typen von Asynchron-Nachrichten:

– Bei Asynchron-Nachrichten an eine *UTM-gesteuerte Queue* wird die Weiterverarbeitung komplett durch *open*UTM gesteuert. Zu diesem Typ gehören Nachrichten, die einen lokalen oder fernen *Asynchron-Vorgang* starten (vgl. auch *Hintergrundauftrag*) und Nachrichten, die zur Ausgabe an ein Terminal oder einen Drucker geschickt wird (vgl. auch *Ausgabeauftrag*).
– Bei Asynchron-Nachrichten an eine *Service-gesteuerte Queue* wird die Weiterverarbeitung durch einen *Service* der Anwendung gesteuert. Zu diesem Typ gehören Nachrichten an eine *TAC-Queue*, Nachrichten an eine *USER-Queue* und Nachrichten an eine *Temporäre Queue*. Die User-Queue und die Temporäre Queue müssen dabei zur lokalen Anwendung gehören, die TAC-Queue kann sowohl in der lokalen als auch in einer fernen Anwendung liegen.

Asynchron-Programm
asynchronous program

Teilprogramm, das von einem *Hintergrundauftrag* gestartet wird.

Asynchron-Vorgang (KDCS)
asynchronous service

Vorgang, der einen *Hintergrundauftrag* bearbeitet. Die Verarbeitung erfolgt entkoppelt vom Auftraggeber. Ein Asynchron-Vorgang kann aus einem oder mehreren Teilprogrammen/Transaktionen bestehen. Er wird über einen asynchronen *Transaktionscode* gestartet.

Auftrag
job

Anforderung eines *Services*, der von einer *UTM-Anwendung* zur Verfügung gestellt wird, durch Angabe eines Transaktionscodes. Siehe auch: *Ausgabeauftrag*, *Dialogauftrag*, *Hintergrundauftrag*, *Auftragskomplex*.

Auftraggeber-Vorgang
job-submitting service

Ein Auftraggeber-Vorgang ist ein *Vorgang*, der zur Bearbeitung eines Auftrags einen Service von einer anderen Server-Anwendung (*Auftragnehmer-Vorgang*) anfordert.

Auftragnehmer-Vorgang
job-receiving service

Ein Auftragnehmer-Vorgang ist ein *Vorgang*, der von einem *Auftraggeber-Vorgang* einer anderen Server-Anwendung gestartet wird.

Auftragskomplex
job complex

Auftragskomplexe dienen dazu, *Asynchronaufträgen Quittungsaufträge* zuzuordnen. Ein Asynchronauftrag innerhalb eines Auftragskomplexes wird *Basisauftrag* genannt.

Ausgabeauftrag
queued output job

Ausgabeaufträge sind *Asynchronaufträge*, die die Aufgabe haben, eine Nachricht, z.B. ein Dokument, an einen Drucker oder an ein Terminal auszugeben. Ausgabeziel kann aber auch eine andere Anwendung sein, die über die Transportsystem-Schnittstelle angeschlossen wurde.
Ausgabeaufträge werden ausschließlich von UTM-Systemfunktionen bearbeitet, d.h. für die Bearbeitung müssen keine Teilprogramme erstellt werden.

Authentisierung
authentication

siehe *Zugangskontrolle*.

Autorisierung
authorization

siehe *Zugriffskontrolle*.

Basisauftrag
basic job

Asynchronauftrag in einem *Auftragskomplex*.

Basisformat
basic format

Format, in das der Terminal-Benutzer alle Angaben eintragen kann, die notwendig sind, um einen Vorgang zu starten.

Benutzerausgang
user exit

Begriff ersetzt durch *Event-Exit*.

Benutzerkennung
user ID

Bezeichner für einen Benutzer, der in der *Konfiguration* der *UTM-Anwendung* festgelegt ist (optional mit Passwort zur *Zugangskontrolle*) und dem spezielle Zugriffsrechte (*Zugriffskontrolle*) zugeordnet sind. Ein Terminal-Benutzer oder ein *Client* muss bei der Anmeldung an die UTM-Anwendung diesen Bezeichner (und ggf. das zugeordnete Passwort) angeben. UTM-Anwendungen können auch ohne Benutzerkennungen generiert werden.

Benutzer-Protokolldatei
user log file

Datei oder Dateigeneration, in die der Benutzer mit dem LPUT-Aufruf Sätze variabler Länge schreibt. Jedem Satz werden die Daten aus dem KB-Kopf des *KDCS-Kommunikationsbereichs* vorangestellt. Die Benutzerprotokolldatei unterliegt der Transaktionssicherung von *open*UTM.

Berechtigungsprüfung
sign-on check

siehe *Zugangskontrolle*.

Beweissicherung (BS2000/OSD)
audit

Im Betrieb einer *UTM-Anwendung* werden zur Beweissicherung sicherheitsrelevante UTM-Ereignisse von *SAT* protokolliert.

Bildschirm-Wiederanlauf
screen restart

Wird ein *Dialog-Vorgang* unterbrochen, gibt *open*UTM beim *Vorgangswiederanlauf* die *Dialog-Nachricht* der letzten abgeschlossenen *Transaktion* erneut auf dem Bildschirm aus.

Browsen von Asynchron-Nachrichten
browsing asynchronous messages

Ein *Vorgang* liest nacheinander die *Asynchron-Nachrichten*, die sich in einer *Service-gesteuerten Queue* befinden. Die Nachrichten werden während des Lesens nicht gesperrt und verbleiben nach dem Lesen in der Queue. Dadurch ist gleichzeitiges Lesen durch unterschiedliche Vorgänge möglich.

Bypass-Betrieb (BS2000/OSD)
bypass mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Im Bypass-Betrieb wird eine an den Drucker gerichtete *Asynchron-Nachricht* an das Terminal gesendet und von diesem auf den Drucker umgeleitet, ohne auf dem Bildschirm angezeigt zu werden.

Cache-Speicher
cache

Pufferbereich zur Zwischenspeicherung von Anwenderdaten für alle Prozesse einer *UTM-Anwendung*. Der Cache-Speicher dient zur Optimierung der Zugriffe auf den *Pagepool*.

CCS-Name (BS2000/OSD)
CCS name

siehe *Coded-Character-Set-Name*.

Client
client

Clients einer *UTM-Anwendung* können sein:
- Terminals
- UPIC-Client-Programme
- Transportsystem-Anwendungen (z.B. DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als *Transportsystem-Anwendung* generiert sind)

Clients werden über LTERM-Partner an die UTM-Anwendung angeschlossen.

*open*UTM-Clients mit Trägersystem OpenCPIC werden wie *OSI-TP-Partner* behandelt.

Client-Seite einer Conversation
client side of a conversation

Begriff ersetzt durch *Initiator*.

Coded-Character-Set-Name (BS2000/OSD)
coded character set name

Bei Verwendung des Produkts *XHCS* (e**X**tended **H**ost **C**ode **S**upport) wird jeder verwendete Zeichensatz durch einen Coded-Character-Set-Namen (abgekürzt: „CCS-Name“ oder „CCSN“) eindeutig identifiziert.

Communication Resource Manager
communication resource manager

Communication Resource Manager (CRMs) kontrollieren in verteilten Systemen die Kommunikation zwischen den Anwendungsprogrammen. *open*UTM stellt CRMs für den internationalen Standard OSI TP, für den Industrie-Standard *LU6.1* und für das *open*UTM-eigene Protokoll UPIC zur Verfügung.

Contention Loser
contention loser

Bei der Kommunikation zwischen zwei Server-Anwendungen über LU6.1 ist der Contention Loser der Partner des *Contention Winner*.

Contention Winner
contention winner

Bei der Kommunikation zwischen zwei Server-Anwendungen über LU6.1 ist diejenige Anwendung der Contention Winner, die die Verbindung verwaltet.

Conversation
conversation

Bei CPI-C nennt man die Kommunikation zwischen zwei CPI-C-Anwendungsprogrammen Conversation. Die Kommunikationspartner einer Conversation werden *Initiator* und *Akzeptor* genannt.

Conversation-ID
conversation ID

Jeder *Conversation* wird von CPI-C lokal eine Conversation-ID zugeordnet, d.h. *Initiator* und *Akzeptor* haben jeweils eine eigene Conversation-ID. Mit der Conversation-ID wird jeder CPI-C-Aufruf innerhalb eines Programms eindeutig einer Conversation zugeordnet.

CPI-C

CPI-C (Common Programming Interface for Communication) ist eine von X/Open und dem CIW (**C**PI-C **I**mplementor's **W**orkshop) normierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen. Das in *open*UTM implementierte CPI-C genügt der CPI-C V2.0 CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. CPI-C in *open*UTM kann über die Protokolle OSI-TP, *LU6.1*, UPIC und mit *open*UTM-LU62 kommunizieren.

DES

DES (Data Encryption Standard) ist eine internationale Norm zur Verschlüsselung von Daten. Bei diesem Verfahren wird ein Schlüssel zum Ver- und Entschlüsseln verwendet. Wird das DES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen DES-Schlüssel.

Dialog-Auftrag
dialog job, interactive job

Auftrag, der einen *Dialog-Vorgang* startet. Der Auftrag kann von einem *Client* oder - bei *Server-Server-Kommunikation* - von einer anderen Anwendung (Client-Vorgang) erteilt werden.

Dialog-Conversation
dialog conversation

CPI-C-Conversation, bei der sowohl der *Initiator* als auch der *Akzeptor* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein Dialog-Transaktionscode generiert sein.

Dialog-Nachricht
dialog message

Nachricht, die eine Antwort erfordert oder selbst eine Antwort auf eine Anfrage ist. Dabei bilden Anfrage und Antwort einen *Dialogschritt*.

Dialogprogramm
dialog program

Teilprogramm, das einen *Dialogschritt* teilweise oder vollständig bearbeitet.

Dialogschritt
dialog step

Ein Dialogschritt beginnt mit dem Empfang einer *Dialog-Nachricht* durch die *UTM-Anwendung*. Er endet mit der Antwort der UTM-Anwendung.

Dialogterminalprozess (UNIX/Windows)
dialog terminal process

Ein Dialogterminalprozess verbindet ein UNIX/Windows-Terminal mit den *Work-prozessen* der *UTM-Anwendung*. Dialogterminalprozesse werden entweder vom Benutzer durch Eingabe von utmdtp oder über die LOGIN-Shell gestartet. Für jedes Terminal, das an eine UTM-Anwendung angeschlossen werden soll, ist ein eigener Dialogterminalprozess erforderlich.

Dialog-Vorgang
dialog service

Vorgang, der einen *Auftrag* im Dialog (zeitlich gekoppelt) mit dem Auftraggeber bearbeitet. Ein Dialog-Vorgang verarbeitet *Dialog-Nachrichten* vom Auftraggeber und erzeugt Dialog-Nachrichten für diesen. Ein Dialog-Vorgang besteht aus mindestens einer *Transaktion.* Ein Dialog-Vorgang umfasst in der Regel mindestens einen *Dialogschritt*. Ausnahme: Bei *Vorgangskettung* können auch mehrere Vorgänge einen Dialogschritt bilden.

Dienst
service

Programm, das im Hintergrund unabhängig von angemeldeten Benutzern oder Fenstern abläuft.

Dienstzugriffspunkt
service access point

Im *OSI-Referenzmodell* stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen *Selektor* identifiziert. Bei der Kommunikation bindet sich die *UTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Transaction Processing

X/Open-Architekturmodell für die transaktionsorientierte *verteilte Verarbeitung*.

Druckadministration
print administration

Funktionen zur *Drucksteuerung* und Administration von *Ausgabeaufträgen*, die an einen Drucker gerichtet sind.

Druckerbündel
printer pool

Mehrere Drucker, die demselben *LTERM-Partner* zugeordnet sind.

Druckergruppe (UNIX)
printer group

UNIX richtet für jeden Drucker standardmäßig eine Druckergruppe ein, die genau diesen Drucker enthält. Darüber hinaus lassen sich mehrere Drucker einer Druckergruppe, aber auch ein Drucker mehreren Druckergruppen zuordnen.

Druckerprozess (UNIX)
printer process

Prozess, der vom *Mainprozess* zur Ausgabe von *Asynchron-Nachrichten* an eine *Druckergruppe* eingerichtet wird. Er existiert, solange die Druckergruppe an die *UTM-Anwendung* angeschlossen ist. Pro angeschlossener Druckergruppe gibt es einen Druckerprozess.

Druckersteuerstation
printer control terminal

Begriff ersetzt durch *Druckersteuer-LTERM*

Druckersteuer-LTERM
printer control LTERM

Über ein Druckersteuer-LTERM kann sich ein *Client* oder ein Terminal-Benutzer an eine *UTM-Anwendung* anschließen. Von dem Client-Programm oder Terminal aus kann dann die *Administration* der Drucker erfolgen, die dem Druckersteuer-LTERM zugeordnet sind. Hierfür ist keine Administrationsberechtigung notwendig.

Drucksteuerung
print control

*open*UTM-Funktionen zur Steuerung von Druckausgaben.

Dynamische Konfigurierung
dynamic configuration

Änderung der *Konfiguration* durch die Administration. Im laufenden Betrieb der Anwendung können UTM-Objekte wie z.B. *Teilprogramme*, *Transaktionscodes*, *Clients*, *LU6.1-Verbindungen*, Drucker oder *Benutzerkennungen* in die Konfiguration aufgenommen, modifiziert oder teilweise auch gelöscht werden. Hierzu kann das Administrationsprogramm WinAdmin verwendet werden, oder es müssen eigene *Administrationsprogramme* erstellt werden, die die Funktionen der *Programmschnittstelle der Administration* nutzen.

Einschritt-Transaktion
single-step transaction

Transaktion, die genau einen *Dialogschritt* umfasst.

Einschritt-Vorgang
single-step service

Dialog-Vorgang, der genau einen *Dialogschritt* umfasst.

Ereignisgesteuerter Vorgang
event-driven service

Begriff ersetzt durch *Event-Service*.

Event-Exit
event exit

Routine des *Anwendungsprogramms*, das bei bestimmten Ereignissen (z.B. Start eines Prozesses, Ende eines Vorgangs) automatisch gestartet wird. Diese darf - im Gegensatz zu den *Event-Services* - keine KDCS-, CPI-C- und XATMI-Aufrufe enthalten.

Event-Funktion
event function

Oberbegriff für *Event-Exits* und *Event-Services*.

Event-Service
event service

Vorgang, der beim Auftreten bestimmter Ereignisse gestartet wird, z.B. bei bestimmten UTM-Meldungen. Die *Teilprogramme* ereignisgesteuerter Vorgänge müssen KDCS-Aufrufe enthalten.

Generierung
generation

Statische Konfigurierung einer *UTM-Anwendung* mit dem UTM-Tool KDCDEF und Erzeugen des Anwendungsprogramms.
Bei der Generierung entstehen die *KDCFILE* und die Source für das Tabellenmodul der Main-Routine *KDCROOT*.

Globaler Name

Symbolischer Name, den *TNS* zur Adressierung einer Anwendung benötigt. Unter dem globalen Namen werden die Adressen der lokalen als auch fernen Anwendungen in der TNS-Datenbasis abgelegt. Der globale Name der lokalen Anwendung ist der Anwendungsname (BCAMAPPL-Name, Name des OSI-TP-Zugriffspunkts). Der globale Name einer fernen Anwendung setzt sich zusammen aus Stationsname und Prozessorname.

Globaler Sekundärer Speicherbereich/GSSB
global secondary storage area

siehe *Sekundärspeicherbereich*.

Globale Transaktion
global transaction

Transaktion, die sich über mehr als eine Anwendung erstreckt.

Hardcopy-Betrieb
hardcopy mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Dabei wird eine Nachricht, die auf dem Bildschirm angezeigt wird, zusätzlich auf dem Drucker abgedruckt.

Heterogene Kopplung
heterogeneous link

Bei *Server-Server-Kommunikation:* Kopplung einer *UTM-Anwendung* mit einer Nicht-UTM-Anwendung, z.B. einer CICS- oder TUXEDO-Anwendung.

Hintergrundauftrag
background job

Hintergrund-Aufträge sind *Asynchron-Aufträge*, die an einen *Asynchron-Vorgang* der eigenen oder einer fernen Anwendung gerichtet sind. Hintergrund-Aufträge eignen sich besonders für zeitintensive oder zeitunkritische Verarbeitungen, deren Ergebnis keinen direkten Einfluss auf den aktuellen Dialog hat.

Homogene Kopplung
homogeneous link

Bei *Server-Server-Kommunikation*: Kopplung von *UTM-Anwendungen.* Dabei spielt es keine Rolle, ob die Anwendungen auf der gleichen oder auf unterschiedlichen Betriebssystem-Plattformen ablaufen.

Inbound-Conversation (CPI-C)
inbound conversation

siehe *Incoming-Conversation.*

Incoming-Conversation (CPI-C)
incoming conversation

Eine *Conversation*, bei der das lokale CPI-C-Programm *Akzeptor* ist, heißt Incoming-Conversation. In der X/Open-Specification wird für Incoming-Conversation auch das Synonym Inbound-Conversation verwendet.

Initiator (CPI-C)
initiator

Die Kommunikationspartner einer *Conversation* werden Initiator und *Akzeptor* genannt. Der Initiator baut die Conversation mit den CPI-C-Aufrufen Initialize_Conversation und Allocate auf.

Insert
insert

Feld in einem Meldungstext, in das *open*UTM aktuelle Werte einträgt.

Inverser KDCDEF
inverse KDCDEF

Funktion, die aus den Konfigurationsdaten der *KDCFILE*, die im laufenden Betrieb dynamisch angepasst wurde, Steueranweisungen für einen *KDCDEF*-Lauf erzeugt. Der inverse KDCDEF kann „offline“ unter KDCDEF oder „online“ über die *Programmschnittstelle zur Administration* gestartet werden.

Kaltstart
cold start

Starten einer *UTM-Anwendung* nach einer *normalen Beendigung* der Anwendung oder nach einer Neugenerierung (vgl. auch *Warmstart*).

KDCADM

Standard-Administrationsprogramm, das zusammen mit *open*UTM ausgeliefert wird. KDCADM stellt Administrationsfunktionen zur Verfügung, die über Transaktionscodes (*Administrationskommandos*) aufgerufen werden.

KDCDEF

UTM-Tool für die *Generierung* von *UTM-Anwendungen*. KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die UTM-Objekte *KDCFILE* und die ROOT-Tabellen-Source für die Main Routine *KDCROOT*.

KDCFILE

Eine oder mehrere Dateien, die für den Ablauf einer *UTM-Anwendung* notwendige Daten enthalten. Die KDCFILE wird mit dem UTM-Generierungstool *KDCDEF* erstellt. Die KDCFILE enthält unter anderem die *Konfiguration* der Anwendung.

KDCROOT

Main Routine eines *Anwendungsprogramms*, die das Bindeglied zwischen *Teilprogrammen* und UTM-Systemcode bildet. KDCROOT wird zusammen mit den *Teilprogrammen* zum *Anwendungsprogramm* gebunden.

KDCS-Parameterbereich

KDCS parameter area

siehe *Parameterbereich*.

KDCS-Programmschnittstelle

KDCS program interface

Universelle UTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen und *Message Queuing* Funktionen nutzen. Außerdem stellt KDCS Aufrufe zur *verteilten Verarbeitung* zur Verfügung.

Keycode

key code

Code, der in einer Anwendung eine bestimmte Zugriffsberechtigung oder eine bestimmte Rolle repräsentiert. Mehrere Keycodes werden zu einem *Keyset* zusammengefasst.

Keyset
key set

Zusammenfassung von einem oder mehrerer *Keycodes* unter einem bestimmten Namen. Ein Keyset definiert Berechtigungen im Rahmen des verwendeten Berechtigungskonzepts (Lock-/Keycode-Konzept oder *Access-List*-Konzept). Ein Keyset kann einer *Benutzerkennung*, einem *LTERM-Partner*, einem *Service* oder einer *TAC-Queue* zugeordnet werden.

Kommunikationsbereich/KB (KDCS)
communication area

Transaktionsgesicherter KDCS-*Primärspeicherbereich*, der vorgangsspezifische Daten enthält. Der Kommunikationsbereich besteht aus 3 Teilen:

– dem KB-Kopf mit allgemeinen Vorgangsdaten,
– dem KB-Rückgabebereich für Rückgaben nach KDCS-Aufrufen
– dem KB-Programmbereich zur Datenübergabe zwischen UTM-Teilprogrammen innerhalb eines *Vorgangs*.

Konfiguration
configuration

Summe aller Eigenschaften einer *UTM-Anwendung*. Die Konfiguration beschreibt:

– Anwendungs- und Betriebsparameter
– die Objekte der Anwendung und die Eigenschaften dieser Objekte. Objekte sind z.B. *Teilprogramme* und *Transaktionscodes*, Kommunikationspartner, Drucker, *Benutzerkennungen*
– definierte Zugriffsschutz- und Zugangsschutzmaßnahmen

Die Konfiguration einer UTM-Anwendung wird bei der Generierung festgelegt und kann per *Administration* dynamisch (während des Anwendungslaufs) geändert werden. Die Konfiguration ist in der *KDCFILE* abgelegt.

Konfigurierung
configuration

Festlegen der *Konfiguration* der UTM-Anwendung. Es wird unterschieden zwischen *statischer* und *dynamischer Konfigurierung*.

Logische Verbindung
virtual connection

Zuordnung zweier Kommunikationspartner.

Lockcode

Code, um einen LTERM-Partner oder einen Transaktionscode vor unberechtigtem Zugriff zu schützen. Damit ist ein Zugriff nur möglich, wenn das *Keyset* des Zugreifenden den passenden *Keycode* enthält (Lock-/Keycode-Konzept).

Lokaler Sekundärer Speicherbereich/LSSB
local secondary storage area
siehe *Sekundärspeicherbereich.*

LPAP-Partner
LPAP partner
Für die *verteilte Verarbeitung* über das *LU6.1*-Protokoll muss in der lokalen Anwendung für jede Partneranwendung ein LPAP-Partner konfiguriert werden. Der LPAP-Partner spiegelt in der lokalen Anwendung die Partneranwendung wider. Bei der Kommunikation wird die Partneranwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten LPAP-Partners angesprochen.

LTERM-Partner
LTERM partner
Um *Clients* oder Drucker an eine *UTM-Anwendung* anschließen zu können, müssen in der Anwendung LTERM-Partner konfiguriert werden. Ein Client oder Drucker kann nur angeschlossen werden, wenn ihm ein LTERM-Partner mit entsprechenden Eigenschaften zugeordnet ist. Diese Zuordnung wird i.A. bei der Konfigurierung festgelegt, sie kann aber auch dynamisch über Terminal-Pools erfolgen.

LTERM-Pool
LTERM pool
Statt für jeden *Client* eine LTERM- und eine PTERM-Anweisung anzugeben, kann mit der Anweisung TPOOL ein Pool von LTERM-Partnern definiert werden. Schließt sich ein Client über einen LTERM-Pool an, wird ihm dynamisch ein LTERM-Partner aus dem Pool zugeordnet.

LU6.1
Geräteunabhängiges Datenaustauschprotokoll (Industrie-Standard) für die transaktionsgesicherte *Server-Server-Kommunikation*.

Mainprozess (UNIX/Windows)
main process
Prozess, der die *UTM-Anwendung* startet. Er startet außerdem die *Workprozesse*, die *Druckerprozesse, Netzprozesse* und den *Timerprozess* und überwacht und steuert die *UTM-Anwendung*.

Main Routine KDCROOT
main routine KDCROOT
siehe *KDCROOT.*

Meldung
UTM message

Meldungen werden von dem Transaktionsmonitor *open*UTM oder von UTM-Tools (wie z.B. *KDCDEF*) an *Meldungsziele* ausgegeben. Eine Meldung besteht aus einer Meldungsnummer und dem Meldungstext, der ggf. *Inserts* mit aktuellen Werten enthält. Je nach Meldungsziel werden entweder die gesamte Meldung oder nur Teile der Meldung (z.B. nur die Inserts) ausgegeben.

Meldungsdefinitionsdatei
UTM message definition file

Die Meldungsdefinitionsdatei wird mit *open*UTM ausgeliefert und enthält standardmäßig die UTM-Meldungstexte in deutscher und englischer Sprache und die Definitionen der Meldungseigenschaften. Aufbauend auf diese Datei kann der Anwender auch eigene, individuelle Meldungsmodule erzeugen.

Meldungsziel
UTM message destination

Ausgabemedium für eine *Meldung*. Mögliche Meldungsziele einer Meldung sind z.B. Terminals, *TS-Anwendungen*, der *Event-Service* MSGTAC, die *System-Protokolldatei* SYSLOG oder *TAC-Queues, Asynchron-TACs* und *USER-Queues*.

Mehrschritt-Transaktion
multi-step transaction

Transaktion, die aus mehr als einem *Verarbeitungsschritt* besteht.

Mehrschritt-Vorgang (KDCS)
multi-step service

Vorgang, der in mehreren *Dialogschritten* ausgeführt wird.

Message Queuing
message queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete *Message Queues* ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen, die Übermittlung der Nachricht wird garantiert, unabhängig davon, ob gerade eine Netzverbindung besteht oder nicht. Bei *open*UTM gibt es *UTM-gesteuerte Queues* und *Service-gesteuerte Queues*.

Message Queue
message queue

Warteschlange, in der bestimmte Nachrichten transaktionsgesichert bis zur Weiterverarbeitung eingereiht werden. Je nachdem, wer die Weiterverarbeitung kontrolliert, unterscheidet man *Service-gesteuerte Queues* und *UTM-gesteuerte Queues*.

MSGTAC
MSGTAC

Spezieller Event-Service, der Meldungen mit dem Meldungsziel MSGTAC per Programm verarbeitet. MSGTAC ist ein Asynchron-Vorgang und wird vom Betreiber der Anwendung erstellt.

Multiplexanschluss (BS2000/OSD)
multiplex connection

Spezielle Möglichkeit, Terminals an eine *UTM-Anwendung* anzuschließen. Ein Multiplexanschluss ermöglicht es, dass sich mehrere Terminals eine *Transportverbindung* teilen.

Nachrichtenbereich/NB (KDCS)
KDCS message area

Bei KDCS-Aufrufen: Puffer-Bereich, in dem Nachrichten oder Daten für *open*UTM oder für das *Teilprogramm* bereitgestellt werden.

Nachrichtenverteiler (BS2000/OSD)
message router

Einrichtung in einem zentralen Rechner oder Kommunikationsrechner zur Verteilung von Eingabenachrichten an unterschiedliche *UTM-Anwendungen*, die auf unterschiedlichen Rechnern liegen können. Der Nachrichtenverteiler ermöglicht außerdem, mit *Multiplexanschlüssen* zu arbeiten.

Netzprozess (UNIX/Windows)
net process

Prozess einer *UTM-Anwendung* zur Netzanbindung.

Netzwerk-Selektor
network selector

Der Netzwerk-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Vermittlungsschicht des *OSI-Referenzmodells*.

Normale Beendigung einer UTM-Anwendung
normal termination of a UTM application

Kontrollierte Beendigung einer *UTM-Anwendung*; das bedeutet u.a., dass die Verwaltungsdaten auf der *KDCFILE* aktualisiert werden. Eine normale Beendigung veranlasst der *Administrator* (z.B. mit KDCSHUT N). Den Start nach einer normalen Beendigung führt *open*UTM als *Kaltstart* durch.

Object Identifier
object identifier

Ein Object Identifier ist ein global (d.h. weltweit) eindeutiger Bezeichner für Objekte im OSI-Umfeld. Ein Object Identifier besteht aus einer Folge von ganzen Zahlen, die einen Pfad in einer Baumstruktur repräsentiert.

Offener Terminalpool
open terminal pool

Terminalpool, der nicht auf *Clients* eines Rechners oder eines bestimmten Typs beschränkt ist. An diesen Terminalpool können sich alle Clients anschließen, für die kein rechner- oder typspezifischer Terminalpool generiert ist.

***open*UTM-Anwendung**
*open*UTM application

siehe *UTM-Anwendung*.

***open*UTM-D**

*open*UTM-D (*open*UTM-Distributed) ist eine *open*UTM-Komponente, die *verteilte Verarbeitung* ermöglicht. *open*UTM-D ist integraler Bestandteil von *open*UTM.

OSI-LPAP-Partner
OSI-LPAP partner

OSI-LPAP-Partner sind die bei *open*UTM generierten Adressen der *OSI-TP-Partner*. Für die *verteilte Verarbeitung* über das Protokoll *OSI TP* muss in der lokalen Anwendung für jede Partneranwendung ein OSI-LPAP-Partner konfiguriert werden. Der OSI-LPAP-Partner spiegelt in der lokalen Anwendung die Partneranwendung wider. Bei der Kommunikation wird die Partneranwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell
OSI reference model

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Von der ISO definiertes Kommunikationsprotokoll für die verteilte Transaktionsverarbeitung. OSI TP steht für Open System Interconnection Transaction Processing.

OSI-TP-Partner
OSI-TP partner

Partner der UTM-Anwendung, der mit der UTM-Anwendung über das OSI-TP-Protokoll kommuniziert.
Beispiele für solche Partner sind:
- eine UTM-Anwendung, die über OSI TP kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS), die über *open*UTM-LU62 angeschlossen ist
- eine Anwendung des Trägersystems OpenCPIC des *open*UTM-Client
- Anwendungen anderer TP-Monitore, die OSI TP unterstützen

Outbound-Conversation (CPI-C)
outbound conversation

siehe *Outgoing-Conversation*.

Outgoing-Conversation (CPI-C)
outgoing conversation

Eine Conversation, bei der das lokale CPI-C-Programm der *Initiator* ist, heißt Outgoing-Conversation. In der X/Open-Specification wird für Outgoing-Conversation auch das Synonym Outbound-Conversation verwendet.

Pagepool
page pool

Teil der *KDCFILE*, in dem Anwenderdaten gespeichert werden, z.B. *Dialog-Nachrichten*, Nachrichten an *Message Queues*, *Sekundärspeicherbereiche*.

Parameterbereich
parameter area

Datenstruktur, in der ein *Teilprogramm* bei einem UTM-Aufruf die für diesen Aufruf notwendigen Operanden an *open*UTM übergibt.

Postselection (BS2000/OSD)
postselection

Auswahl der protokollierten UTM-Ereignisse aus der SAT-Protokolldatei, die ausgewertet werden sollen. Die Auswahl erfolgt mit Hilfe des Tools SATUT.

Prädialog (BS2000/OSD)
predialog

Aufforderung des Terminal-Benutzers an das Datenkommunikationssystem zum Aufbau einer *logischen Verbindung* zur *Anwendung*. Der Prädialog entfällt, wenn die logische Verbindung auf Initiative der Anwendung aufgebaut wird.

Preselection (BS2000/OSD)
preselection

Festlegung der für die *SAT-Beweissicherung* zu protokollierenden UTM-Ereignisse. Die Preselection erfolgt durch die UTM-SAT-Administration. Man unterscheidet ereignisspezifische, benutzerspezifische und auftrags-(TAC-)spezifische Preselection.

Presentation-Selektor
presentation selector

Der Presentation-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Darstellungsschicht des *OSI-Referenzmodells*.

Primärspeicherbereich
primary storage area

Bereich im Arbeitsspeicher, auf den das *KDCS-Teilprogramm* direkt zugreifen kann, z.B. *Standard Primärer Arbeitsbereich*, *Kommunikationsbereich*.

Printerprozess (UNIX)
printer process

siehe *Druckerprozess*.

Programmschnittstelle zur Administration
program interface for administration

UTM-Programmschnittstelle, mit deren Hilfe der Anwender eigene *Administrationsprogramme* erstellen kann. Die Programmschnittstelle zur Administration bietet u.a. Funktionen zur *dynamischen Konfigurierung*, zur Modifikation von Eigenschaften und Anwendungsparametern und zur Abfrage von Informationen zur *Konfiguration* und zur aktuellen Auslastung der Anwendung.

Prozess
process

In den *open*UTM-Handbüchern wird der Begriff „Prozess“ als Oberbegriff für Prozess (UNIX/Windows) und Task (BS2000/OSD) verwendet.

Queue
queue

siehe *Message Queue*

QuickStart Kit

Beispielanwendung, die mit *open*UTM (Windows) ausgeliefert wird.

Quittungsauftrag
confirmation job

Bestandteil eines *Auftragskomplexes*, worin der Quittungsauftrag dem *Basisauftrag* zugeordnet ist. Es gibt positive und negative Quittungsaufträge. Bei positivem Ergebnis des *Basisauftrags* wird der positive Quittungsauftrag wirksam, sonst der negative.

Redelivery
redelivery

Erneutes Zustellen einer *Asynchron-Nachricht*, nachdem diese nicht ordnungsgemäß verarbeitet werden konnte, z.B. weil die *Transaktion* zurückgesetzt oder der *Asynchron-Vorgang* abnormal beendet wurde. Die Nachricht wird wieder in die Message Queue eingereiht und lässt sich damit erneut lesen und/oder verarbeiten.

Reentrant-fähiges Programm
reentrant program

Programm, dessen Code durch die Ausführung nicht verändert wird.
Im BS2000/OSD ist dies Voraussetzung dafür, *Shared Code* zu nutzen.

Request
request

Anforderung einer *Service-Funktion* durch einen *Client* oder einen anderen Server.

Requestor
requestor

In XATMI steht der Begriff Requestor für eine Anwendung, die einen Service aufruft.

Resource Manager
resource manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme. *open*UTM stellt aber auch selbst Resource Manager zur Verfügung, z.B. für den Zugriff auf *Message Queues*, lokale Speicherbereiche und Logging-Dateien. Anwendungsprogramme greifen auf RMs über RM-spezifische Schnittstellen zu. Für Datenbank-Systeme ist dies meist SQL, für die *open*UTM-RMs die Schnittstelle KDCS.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

RSA

Abkürzung für die Erfinder dieses Verschlüsselungsverfahrens Rivest, Shamir und Adleman. Bei diesem Verfahren wird ein Schlüsselpaar verwendet, das aus einem öffentlichen und einem privaten Schlüssel besteht. Eine Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und kann nur mit dem privaten Schlüssel entschlüsselt werden. Das RSA-Schlüsselpaar wird von der UTM-Anwendung erzeugt.

SAT-Beweissicherung (BS2000/OSD)

SAT audit

Beweissicherung durch die Komponente SAT (Security Audit Trail) des BS2000-Softwareproduktes SECOS.

Sekundärspeicherbereich

secondary storage area

Transaktionsgesicherter Speicherbereich auf der *KDCFILE*, auf den das KDCS-*Teilprogramm* mit speziellen Aufrufen zugreifen kann. Lokale Sekundärspeicherbereiche (LSSB) sind einem *Vorgang* zugeordnet, auf globale Sekundärspeicherbereiche (GSSB) kann von allen Vorgängen einer *UTM-Anwendung* zugegriffen werden. Weitere Sekundärspeicherbereiche sind der *Terminalspezifische Langzeitspeicher (TLS)* und der *Userspezifische Langzeitspeicher (ULS)*.

Selektor

selector

Ein Selektor identifiziert im lokalen System einen *Zugriffspunkt* auf die Dienste einer Schicht des *OSI-Referenzmodells*. Jeder Selektor ist Bestandteil der Adresse des Zugriffspunktes.

Semaphor (UNIX/Windows)

semaphore

UNIX- und Windows-Betriebsmittel, das zur Steuerung und Synchronisation von Prozessen dient.

Server

server

Ein Server ist eine *Anwendung*, die *Services* zur Verfügung stellt. Oft bezeichnet man auch den Rechner, auf dem Server-Anwendungen laufen, als Server.

Server-Seite einer Conversation (CPI-C)

server side of a conversation

Begriff ersetzt durch *Akzeptor*.

Server-Server-Kommunikation
server-server communication
siehe *verteilte Verarbeitung*.

Service Access Point
siehe *Dienstzugriffspunkt.*

Service
service
Services sind Dienstleistungen, die von *Servern* erbracht werden. Diese Services können von *Clients* oder anderen Servern aufgerufen werden.

Service-gesteuerte Queue
service controlled queue
Message Queue, bei der der Abruf und die Weiterverarbeitung der Nachrichten durch *Services* gesteuert werden. Ein Service muss zum Lesen der Nachricht explizit einen KDCS-Aufruf (DGET) absetzen.
Service-gesteuerte Queues gibt es bei *open*UTM in den Varianten *USER-Queue*, *TAC-Queue* und *Temporäre Queue*.

Service Routine
service routine
siehe *Teilprogramm*.

Session
session
Kommunikationsbeziehung zweier adressierbarer Einheiten im Netz über das SNA-Protokoll *LU6.1*.

Session-Selektor
session selector
Der Session-Selektor identifiziert im lokalen System einen *Zugriffspunkt* zu den Diensten der Kommunikationssteuerschicht (Session-Layer) des *OSI-Referenzmodells*.

Shared Code (BS2000/OSD)
shared code
Code, der von mehreren Prozessen gleichzeitig benutzt werden kann.

Shared Memory
shared memory
Virtueller Speicherbereich, auf den mehrere Prozesse gleichzeitig zugreifen können.

Shared Objects (UNIX/Windows)
shared objects

Teile des *Anwendungsprogramms* können als Shared Objects erzeugt werden. Diese werden dynamisch zur Anwendung dazugebunden und können im laufenden Betrieb ausgetauscht werden. Shared Objects werden mit der KDCDEF-Anweisung SHARED-OBJECT definiert.

Sicherungspunkt
synchronization point, consistency point

Ende einer *Transaktion*. Zu diesem Zeitpunkt werden alle in der Transaktion vorgenommenen Änderungen der *Anwendungsinformation* gegen Systemausfall gesichert und für andere sichtbar gemacht. Während der Transaktion gesetzte Sperren werden wieder aufgehoben.

Socket-Verbindung
socket connection

Transportsystem-Verbindung, die die Socket-Schnittstelle verwendet. Die Socket-Schnittstelle ist eine Standard-Programmschnittstelle für die Kommunikation über TCP/IP.

Standard Primärer Arbeitsbereich/SPAB (KDCS)
standard primary working area

Bereich im Arbeitsspeicher, der jedem KDCS-*Teilprogramm* zur Verfügung steht. Sein Inhalt ist zu Beginn des Teilprogrammlaufs undefiniert oder mit einem Füllzeichen vorbelegt.

Startformat
start format

Format, das *open*UTM am Terminal ausgibt, wenn sich ein Benutzer erfolgreich bei der *UTM-Anwendung* angemeldet hat (ausgenommen nach *Vorgangswiederanlauf* und beim *Anmeldevorgang*).

statische Konfigurierung
static configuration

Festlegen der *Konfiguration* bei der Generierung mit Hilfe des UTM-Tools *KDCDEF*.

SYSLOG-Datei
SYSLOG file

siehe *System-Protokolldatei*.

System-Protokolldatei
system log file
Datei oder Dateigeneration, in die *open*UTM während des Laufs einer *UTM-Anwendung* alle UTM-Meldungen protokolliert, für die das *Meldungsziel* SYSLOG definiert ist.

TAC
TAC
siehe *Transaktionscode*.

TAC-Queue
TAC queue
Message Queue, die explizit per KDCDEF-Anweisung generiert wird. Eine TAC-Queue ist eine *Service-gesteuerte Queue* und kann unter dem generierten Namen von jedem Service aus angesprochen werden.

Teilprogramm
program unit
UTM-*Services* werden durch ein oder mehrere Teilprogramme realisiert. Die Teilprogramme sind Bestandteile des *Anwendungsprogramms*. Sie enthalten UTM-Funktionsaufrufe und sind über *Transaktionscodes* ansprechbar. Einem Teilprogramm können mehrere Transaktionscodes zugeordnet werden.

Temporäre Queue
temporary queue
Message Queue, die dynamisch per Programm erzeugt wird und auch wieder per Programm gelöscht werden kann, vgl. *Service-gesteuerte Queue*.

Terminalspezifischer Langzeitspeicher/TLS (KDCS)
terminal-specific long-term storage
Sekundärspeicher, der einem *LTERM-, LPAP-* oder *OSI-LPAP-Partner* zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

Timerprozess (UNIX/Windows)
timer process
Prozess, der Aufträge zur Zeitüberwachung von *Workprozessen* entgegennimmt, sie in ein Auftragsbuch einordnet und nach einer im Auftragsbuch festgelegten Zeit den Workprozessen zur Bearbeitung wieder zustellt.

TNS (UNIX/Windows)
Abkürzung für den Transport Name Service, der einem Anwendungsnamen einen Transport-Selektor und das Transportsystem zuordnet, über das die Anwendung erreichbar ist.

Transaktion
transaction

Verarbeitungsabschnitt innerhalb eines *Services*, der die *ACID-Eigenschaften* aufweist. Von den in einer Transaktion beabsichtigten Änderungen der *Anwendungsinformation* werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt (Alles-oder-Nichts Regel). Das Transaktionsende bildet einen *Sicherungspunkt*.

Transaktionscode/TAC
transaction code

Name, über den ein *Teilprogramm* aufgerufen werden kann. Der Transaktionscode wird dem Teilprogramm bei der *statischen* oder *dynamischen Konfigurierung* zugeordnet. Einem Teilprogramm können auch mehrere Transaktionscodes zugeordnet werden.

Transaktionsrate
transaction rate

Anzahl der erfolgreich beendeten *Transaktionen* pro Zeiteinheit.

Transfer-Syntax
transfer syntax

Bei *OSI TP* werden die Daten zur Übertragung zwischen zwei Rechnersystemen von der lokalen Darstellung in die Transfer-Syntax umgewandelt. Die Transfer-Syntax beschreibt die Daten in einem neutralen Format, das von allen beteiligten Partnern verstanden wird. Jeder Transfer-Syntax muss ein *Object Identifier* zugeordnet sein.

Transport-Selektor
transport selector

Der Transport-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Transportschicht des *OSI-Referenzmodells*.

Transportsystem-Anwendung
transport system application

Anwendung, die direkt auf der Transportsystem-Schnittstelle wie z.B. CMX oder Socket aufsetzt. Für den Anschluss von Transportsystem-Anwendungen muss bei der Konfigurierung als Partnertyp APPLI oder SOCKET angegeben werden. Eine Transportsystem-Anwendung kann nicht in eine globale Transaktionssicherung eingebunden werden.

TS-Anwendung
TS application

siehe *Transportsystem-Anwendung*.

Typisierter Puffer (XATMI)
typed buffer

Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten den Partnern implizit bekannt.

UPIC

Trägersystem für *open*UTM-Clients. UPIC steht für Universal Programming Interface for Communication.

UPIC-Client

Bezeichnung für *open*UTM-Clients mit Trägersystem UPIC

USER-Queue
USER queue

Message Queue, die *open*UTM jeder Benutzerkennung zur Verfügung stellt. Eine USER-Queue zählt zu den *Service-gesteuerten Queues* und ist immer der jeweiligen Benutzerkennung zugeordnet. Der Zugriff von fremden UTM-Benutzern auf die eigene USER-Queue kann eingeschränkt werden.

Userspezifischer Langzeitspeicher/ULS
user-specific long-term storage

Sekundärspeicher, der einer *Benutzerkennung,* einer *Session* oder einer *Association* zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

USLOG-Datei
USLOG file

siehe *Benutzer-Protokolldatei.*

UTM-Anwendung
UTM application

Eine UTM-Anwendung stellt *Services* zur Verfügung, die Aufträge von *Clients* oder anderen *Server*-Anwendungen bearbeiten. *open*UTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine UTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Server-Einheit bildet.

UTM-D

siehe *openUTM-D.*

UTM-Datenstation
UTM terminal

Begriff ersetzt durch *LTERM-Partner*.

UTM-F

UTM-Anwendungen können als UTM-F-Anwendungen (UTM-Fast) generiert werden. Bei UTM-F wird zugunsten der Performance auf Platteneingaben/-ausgaben verzichtet, mit denen bei *UTM-S* die Sicherung von Benutzer- und Transaktionsdaten durchgeführt wird. Gesichert werden lediglich Änderungen der Verwaltungsdaten.

UTM-gesteuerte Queues

UTM controlled queue

Message Queues, bei denen der Abruf und die Weiterverarbeitung der Nachrichten vollständig durch *open*UTM gesteuert werden. Siehe auch *Asynchron-Auftrag, Hintergrundauftrag* und *Asynchron-Nachricht*.

UTM-S

Bei UTM-S-Anwendungen sichert *open*UTM neben den Verwaltungsdaten auch alle Benutzerdaten über ein Anwendungsende und einen Systemausfall hinaus. Außerdem garantiert UTM-S bei allen Störungen die Sicherheit und Konsistenz der Anwendungsdaten. Im Standardfall werden UTM-Anwendungen als UTM-S-Anwendungen (UTM-Secure) generiert.

UTM-SAT-Administration (BS2000/OSD)

UTM SAT administration

Durch die UTM-SAT-Administration wird gesteuert, welche sicherheitsrelevanten UTM-Ereignisse, die im Betrieb der *UTM-Anwendung* auftreten, von *SAT* protokolliert werden sollen. Für die UTM-SAT-Administration wird eine besondere Berechtigung benötigt.

UTM-Seite

UTM page

Ist eine Speichereinheit, die entweder 2K oder 4K umfasst. Die Größe wird bei der Generierung der *UTM-Anwendung* festgelegt. *Pagepool* und Wiederanlauf-Bereich der KDCFILE werden in Einheiten der Größe einer UTM-Seite unterteilt.

Verarbeitungsschritt

processing step

Ein Verarbeitungsschritt beginnt mit dem Empfangen einer *Dialog-Nachricht*, die von einem *Client* oder einer anderen Server-Anwendung an die *UTM-Anwendung* gesendet wird. Der Verarbeitungsschritt endet entweder mit dem Senden einer Antwort und beendet damit auch den *Dialogschritt* oder er endet mit dem Senden einer Dialog-Nachricht an einen Dritten.

Verschlüsselungsstufe
encryption level
: Die Verschlüsselungsstufe legt fest, ob und inwieweit ein Client Nachrichten und Passwort verschlüsseln muss.

Verteilte Transaktion
distributed transaction
: siehe *globale Transaktion*.

Verteilte Transaktionsverarbeitung
Distributed Transaction Processing
: *Verteilte Verarbeitung* mit *globalen Transaktionen*.

Verteilte Verarbeitung
distributed processing
: Bearbeitung von *Dialogaufträgen* durch mehrere Anwendungen oder Übermittlung von *Hintergrundaufträgen* an eine andere Anwendung. Für die verteilte Verarbeitung werden die höheren Kommunikationsprotokolle *LU6.1* und *OSI TP* verwendet. Über *open*UTM-LU62 ist verteilte Verarbeitung auch mit LU6.2 Partnern möglich. Man unterscheidet verteilte Verarbeitung mit *globalen Transaktionen* (anwendungsübergreifende Transaktionssicherung) und verteilte Verarbeitung ohne globale Transaktionen (nur lokale Transaktionssicherung). Die verteilte Verarbeitung wird auch Server-Server-Kommunikation genannt.

Vorgang (KDCS)
service
: Ein Vorgang dient zur Bearbeitung eines *Auftrags* in einer *UTM-Anwendung*. Er setzt sich aus einer oder mehreren *Transaktionen* zusammen. Die erste Transaktion wird über den *Vorgangs-TAC* aufgerufen. Es gibt *Dialog-Vorgänge* und *Asynchron-Vorgänge*. *open*UTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung. Anstelle des Begriffs Vorgang wird häufig auch der allgemeinere Begriff *Service* gebraucht.

Vorgangskellerung (KDCS)
service stacking
: Ein Terminal-Benutzer kann einen laufenden *Dialog-Vorgang* unterbrechen und einen neuen Dialog-Vorgang einschieben. Nach Beendigung des eingeschobenen *Vorgangs* wird der unterbrochene Vorgang fortgesetzt.

Vorgangskettung (KDCS)
service chaining
: Bei Vorgangskettung wird nach Beendigung eines *Dialog-Vorgangs* ohne Angabe einer *Dialog-Nachricht* ein Folgevorgang gestartet.

Vorgangs-TAC (KDCS)
service TAC

Transaktionscode, mit dem ein *Vorgang* gestartet wird.

Vorgangswiederanlauf (KDCS)
service restart

Nach einem Vorgangsabbruch, z.B. infolge Abmeldens des Terminal-Benutzers oder Beendigung der *UTM-Anwendung,* führt *open*UTM einen Vorgangswiederanlauf durch. Ein *Asynchron-Vorgang* wird neu gestartet oder beim zuletzt erreichten *Sicherungspunkt* fortgesetzt, ein *Dialog-Vorgang* wird beim zuletzt erreichten Sicherungspunkt fortgesetzt. Für den Terminal-Benutzer wird der Vorgangswiederanlauf eines Dialog-Vorgangs als *Bildschirm-Wiederanlauf* sichtbar.

Warmstart
warm start

Starten einer *UTM-Anwendung* nach einer *abnormalen Beendigung*. Beim Warmstart wird die *KDCFILE* wieder in einen konsistenten Zustand gebracht.

Wiederanlauf
restart

siehe *Anwendungswiederanlauf*,
siehe *Bildschirm-Wiederanlauf*,
siehe *Vorgangswiederanlauf*.

Workprozess (UNIX/Windows)
work process

Prozess, in dem die *Services* der *UTM-Anwendung* ablaufen.

XATMI

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen.
Das in *open*UTM implementierte XATMI genügt der XATMI CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. XATMI in *open*UTM kann über die Protokolle OSI-TP, *LU6.1* und UPIC kommunizieren.

XHCS (BS2000/OSD)

XHCS (Extended Host Code Support) ist ein BS2000/OSD-Softwareprodukt für die Unterstützung internationaler Zeichensätze.

XML

XML (eXtensible Markup Language) ist eine vom W3C (WWW-Konsortium) genormte Metasprache, in der Austauschformate für Daten und zugehörige Informationen definiert werden können.

Zeitgesteuerter Auftrag
time-driven job

Auftrag, der von *open*UTM bis zu einem definierten Zeitpunkt in einer *Message Queue* zwischengespeichert und dann an den Empfänger weitergeleitet wird. Empfänger kann sein: ein *Asynchron-Vorgang* der selben Anwendung, eine *TAC-Queue*, eine Partneranwendung, ein Terminal oder ein Drucker.
Zeitgesteuerte Aufträge können nur von KDCS-*Teilprogrammen* erteilt werden.

Zugangskontrolle
system access control

Prüfung durch *open*UTM, ob eine bestimmte *Benutzerkennung* berechtigt ist, mit der *UTM-Anwendung* zu arbeiten. Die Berechtigungsprüfung entfällt, wenn die UTM-Anwendung ohne Benutzerkennungen generiert wurde.

Zugriffskontrolle
data access control

Prüfung durch *open*UTM, ob der Kommunikationspartner berechtigt ist, auf ein bestimmtes Objekt der Anwendung zuzugreifen. Die Zugriffsrechte werden als Bestandteil der Konfiguration festgelegt.

Zugriffspunkt
access point

siehe *Dienstzugriffspunkt*.

Abkürzungen

ACSE	Association Control Service Element
AEQ	Application Entity Qualifier
AES	Advanced Encryption Standard
AET	Application Entity Title
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
BCAM	Basic Communication Access Method
BER	Basic Encoding Rules
BLS	Binder-Lader-Starter (BS2000/OSD)
BSI	Bundesamt für Sicherheit in der Informationstechnik
CCP	Communication Control Program
CCR	Commitment, Concurrency and Recovery
CCS	Codierter Zeichensatz (Coded Character Set)
CCSN	Name des codierten Zeichensatzes (Coded Character Set Name)
CICS	Customer Information Control System
CID	Control Identification
CIW	CPI-C Implementor’s Workshop
CMX	Communication Manager in UNIX
COM	Component Object Model
CPI-C	Common Programming Interface for Communication
CRM	Communication Resource Manager
CRTE	Common Runtime Environment (BS2000/OSD)
DB	Datenbank
DC	Data Communication

DCAM	Data Communication Access Method
DCOM	Distributed Component Object Mode
DES	Data Encryption Standard
DMS	Data Management System
DNS	Domain Name Service
DSS	Datensichtstation (=Terminal)
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DVS	Datenverwaltungssystem
EBCDIC	Binärcode für die stellenweise Verschlüsselung von Dezimalziffern (Extended Binary-Coded Decimal Interchange Code)
EJB	Enterprise JavaBeans*TM*
FGG	File Generation Group
FHS	Format Handling System
FORMANT	Format-Manager für Terminals
FT	File Transfer
GSSB	Globaler Sekundärer Speicherbereich
HLL	High-Level Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IFG	Interaktiver Format-Generator
ILCS	Inter Language Communication Services (BS2000/OSD)
IMS	Information Management System (IBM)
IPC	Inter-Process-Communication
IRV	Internationale Referenzversion
ISO	International Organization for Standardization
JCA	Java Connector Architecture
KAA	KDCS Application Area
KB	Kommunikationsbereich
KBPROG	KB-Programmbereich
KDCS	Kompatible Datenkommunikationsschnittstelle
KTA	KDCS Task Area
LAN	Local Area Network

LCF	Local Configuration File
LLM	Link and Load Module (BS2000/OSD)
LSSB	Lokaler Sekundärer Speicherbereich
LU	Logical Unit
MIGRAT	Migrationsprogramm
MQ	Message Queuing
MSCF	Multiple System Control Facility (BS2000/OSD)
MSF	Mehrfachsteuerung für Fernanschluss
MSN	Mehrfachsteuerung für Nahbereich
NEA	Netzwerkarchitektur bei TRANSDATA-Systemen
NB	Nachrichtenbereich
NLS	Unterstützung der Landessprache (Native Language Support)
OCX	OLE Control Extension
OML	Object Modul Library
OLTP	Online Transaction Processing
OSI	Open System Interconnection
OSI-TP	Open System Interconnection Transaction Processing
OSS	OSI Session Service
PCMX	Portable Communication Manager
PDN	Programmsystem für Datenfernverarbeitung und Netzsteuerung
PID	Prozess-Identifikation
PIN	Persönliche Identifikationsnummer
PLU	Primary Logical Unit
RAV	Rechenzentrums-Abrechnungs-Verfahren
RDF	Resource Definition File
RM	Resource Manager
RSA	Encryption-Algorithmus nach Rivest, Shamir, Adleman
RSO	Remote SPOOL Output (BS2000/OSD)
RTS	Runtime System (Laufzeitsystem)
SAT	Security Audit Trail (BS2000/OSD)
SECOS	Security Control System
SGML	Standard Generalized Markup Language

SLU	Secondary Logical Unit
SM2	Software Monitor 2 (BS2000/OSD)
SNA	Systems Network Architecture
SPAB	Standard Primärer Arbeitsbereich
SQL	Structured Query Language
SSB	Sekundärer Speicherbereich
SSO	Single-Sign-On
TAC	Transaktionscode
TCP/IP	Transport Control Protocol / Internet Protocol
TIAM	Terminal Interactive Access Method
TLS	Terminalspezifischer Langzeitspeicher
TM	Transaction Manager
TNS	Transport Name Service
TNSX	Transport Name Service in UNIX / Windows
TP	Transaction Processing (Transaktions-Betrieb)
TPR	Task privileged (privilegierter Funktionszustand des BS2000/OSD)
TPSU	Transaction Protocol Service User
TSN	Task Sequence Number
TU	Task user (nicht privilegierter Funktionszustand des BS2000/OSD)
TX	Transaction Demarcation (X/Open)
UDS	Universelles Datenbanksystem
UDT	Unstructured Data Transfer
ULS	Userspezifischer Langzeitspeicher
UPIC	Universal Programming Interface for Communication
USP	UTM-Socket-Protokoll
UTM	Universeller Transaktionsmonitor
UTM-D	UTM-Funktionen für verteilte Verarbeitung („Distributed“)
UTM-F	Schnelle UTM-Variante („Fast“)
UTM-S	UTM-Sicherheitsvariante
VGID	Vorgangsidentifikation
VTSU	Virtual Terminal Support
VTV	Verteilte Transaktionsverarbeitung
VV	Verteilte Verarbeitung

WAN	Wide Area Network
XA	X/Open Access Interface (Schnittstelle von X/Open zum Zugriff auf Resource Manager)
XAP	X/OPEN ACSE/Presentation programming interface
XAP-TP	X/OPEN ACSE/Presentation programming interface Transaction Processing extension
XATMI	X/Open Application Transaction Manager Interface
XHCS	eXtended Host Code Support
XML	eXtensible Markup Language

Literatur

Zu *open*UTM gibt es auch eine CD-ROM. Diese enthält PDF-Dateien von allen *open*UTM-Handbüchern.
Die Handbücher sind online unter *http://manuals.fujitsu-siemens.com* zu finden oder in gedruckter Form gegen gesondertes Entgelt unter *http://FSC-manualshop.com* zu bestellen.

Dokumentation zu *open*UTM

***open*UTM**
Konzepte und Funktionen
Benutzerhandbuch

Zielgruppe
Alle, die sich einen Überblick über die Funktionsbreite und Leistungsfähigkeit von *open*UTM verschaffen wollen

***open*UTM** (BS2000/OSD, UNIX, Windows)
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

Zielgruppe
Programmierer, die für die Programmierung von UTM-Anwendungen die Programmschnittstelle KDCS nutzen wollen

***open*UTM** (BS2000/OSD, UNIX, Windows)
Anwendungen generieren
Benutzerhandbuch

Zielgruppe
Anwendungsplaner, Anwendungsentwickler und Betreuer von UTM-Anwendungen

***open*UTM** (BS2000/OSD)
Einsatz von *open*UTM-Anwendungen unter BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen

***open*UTM** (BS2000/OSD, UNIX, Windows)
Anwendungen administrieren
Benutzerhandbuch

Zielgruppe
Administratoren und Programmierer von Administrationsprogrammen

***open*UTM** (BS2000/OSD)
Meldungen, Test und Diagnose
Benutzerhandbuch

Zielgruppe
Anwender, Administratoren und Programmierer von UTM-Anwendungen

***open*UTM** (UNIX, Windows)
Meldungen, Test und Diagnose
Benutzerhandbuch

Zielgruppe
Anwender, Administratoren und Programmierer von UTM-Anwendungen

***open*UTM** (BS2000/OSD, UNIX, Windows NT)
Anwendungen erstellen mit X/Open-Schnittstellen
Basishandbuch

Zielgruppe
Programmierer, die für die Programmierung von UTM-Anwendungen die X/Open-Schnittstellen CPI-C, XATMI und TX nutzen wollen

***open*UTM** (BS2000, UNIX, Windows)
***open*UTM Data Marshalling mit XML**
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Programmierer von UTM-Anwendungen

***open*UTM-Client** (UNIX)
für Trägersystem OpenCPIC
Client-Server-Kommunikation mit *open*UTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

***open*UTM-Client V5.2** (UNIX, Windows)
für Trägersystem UPIC
Client-Server-Kommunikation mit *open*UTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

***open*UTM-Client V4.0** (BS2000/OSD, UNIX, Windows)
für Trägersystem UPIC
Client-Server-Kommunikation mit *open*UTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

***open*UTM WinAdmin**
Grafischer Administrationsarbeitsplatz für *open*UTM
Online-Beschreibung (im Internet verfügbar) und Online-Hilfe

Zielgruppe
Organisatoren und Administratoren von UTM-Anwendungen

***open*UTM**, ***open*UTM-LU62** (BS2000/OSD, UNIX, Windows NT)
Verteilte Transaktionsverarbeitung
zwischen *open*UTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer und Administratoren

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-Assembler-Anwendungen

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-Fortran-Anwendungen

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-Pascal-XT-Anwendungen

***open*UTM** (BS2000/OSD)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Programmierer von UTM-PL/I-Anwendungen

***open*UTM**
Masterindex
nur online verfügbar: Internet und *open*UTM-Handbuch-CD

Zielgruppe
Alle, die mit der *open*UTM-Dokumentation arbeiten möchten.

Dokumentation zum openSEAS-Produktumfeld

Bean*Transactions* (Windows, UNIX)
Applikationsserver für Enterprise JavaBeans™

Administrations- und Installationshandbuch

Zielgruppe
Dieses Handbuch richtet sich an Administratoren und Designer von Bean*Transactions*-Anwendungen. Voraussetzungen sind Kenntnisse von Java und der EJB 2.0-Spezifikation von Sun Microsystems.

Bean*Transactions* (Windows, UNIX)
Applikationsserver für Enterprise JavaBeans™

Entwicklungshandbuch

Zielgruppe
Designer von Bean*Transactions*-Anwendungen und Entwickler von Enterprise JavaBeans und Java-Clients.

Biz*Transactions* V3.0
Anwendungsintegration mit Business Objekten
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Dieses Handbuch wendet sich an alle, die mit Business Objekten *open*UTM-, MVS- oder OSD-Anwendungen in Java, Windows oder openSEAS integrieren wollen.

JConnect V2.0
Verbindung von Java-Clients zu *open*UTM
Benutzerdokumentation und Java-Docs
(nur online verfügbar: Internet, *open*UTM-Handbuch-CD)

Zielgruppe
Programmierer von Java-Anwendungen

Web*Transactions*
Konzepte und Funktionen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-, MVS- oder OSD-Anwendungen ans Web anbinden wollen

Web*Transactions*
Template-Sprache
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-, MVS- oder OSD-Anwendungen ans Web anbinden wollen

Web*Transactions*
Anschluss an *open*UTM-Anwendungen über UPIC
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-Anwendungen über UPIC ans Web anbinden wollen

Web*Transactions*
Anschluss an MVS-Anwendungen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die MVS-Anwendungen ans Web anbinden wollen

Web*Transactions*
Anschluss an OSD-Anwendungen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die OSD-Anwendungen ans Web anbinden wollen

Dokumentation zum BS2000/OSD-Umfeld

AID (BS2000/OSD)
Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

Zielgruppe
Programmierer im BS2000

BCAM (BS2000/OSD)
BCAM Band 1/2
Benutzerhandbuch

Zielgruppe
Netzplaner, -generierer und -verwalter, die in BS2000-Systemen BCAM betreiben

BINDER (BS2000/OSD)
Benutzerhandbuch

Zielgruppe
Software-Entwickler

BINDER (BS2000/OSD)
Taschenbuch

Zielgruppe
Software-Entwickler, die im Umgang mit dem BINDER geübt sind

BS2000/OSD
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

Zielgruppe
BS2000/OSD-Assembler-Programmierer

BS2000/OSD-BC
BLSSERV
Bindelader-Starter
Benutzerhandbuch

Zielgruppe
Software-Entwickler und geübte BS2000/OSD-Benutzer

DCAM (BS2000/OSD, TRANSDATA)
COBOL-Aufrufe
Benutzerhandbuch

Zielgruppe
Programmierer von DCAM-COBOL-Programmen

DCAM (BS2000/OSD, TRANSDATA)
Makroaufrufe
Benutzerhandbuch

Zielgruppe
Programmierer von DCAM-Assembler-Programmen

DCAM (BS2000/OSD, TRANSDATA)
Programmschnittstellen
Beschreibung

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer, Systemverwalter und Netzadministratoren

FHS (BS2000/OSD, TRANSDATA)
Formatierungssystem für *open*UTM, TIAM, DCAM
Benutzerhandbuch

Zielgruppe
Programmierer

IFG für FHS (TRANSDATA)
Benutzerhandbuch

Zielgruppe
Datenstationsbenutzer, Anwendungsdesigner und Programmierer

FHS-DOORS (BS2000/OSD, MS-Windows)
Grafische Oberfläche für BS2000/OSD-Anwendungen
Benutzerhandbuch

Zielgruppe
End-Anwender von FHS-DOORS und Administratoren, die Masken konvertieren und diese bereitstellen

HIPLEX AF (BS2000/OSD)
Hochverfügbarkeit von Anwendungen in BS2000/OSD
Produkthandbuch

Zielgruppe
Systemverwalter und Organisatoren in Rechenzentren

IMON (BS2000/OSD)
Installationsmonitor
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an die Systembetreuung des Betriebssystems BS2000/OSD.

MT9750 V5.0 (MS Windows)
9750-Emulation unter Windows
Produkthandbuch

Zielgruppe
Das Handbuch richtet sich an den PC-Benutzer, der mit Hilfe des Produktes MT9750 mit Anwendungen auf einem BS2000-Host kommunizieren will.

OMNIS/OMNIS-MENU (TRANSDATA, BS2000)
Funktionen und Kommandos
Benutzerhandbuch

Zielgruppe
OMNIS-Administrator und OMNIS-Anwender

OMNIS/OMNIS-MENU (TRANSDATA, BS2000)
Administration und Programmierung
Benutzerhandbuch

Zielgruppe
OMNIS-Administrator und Programmierer

OMNIS-MENU (TRANSDATA, BS2000/OSD)
Benutzerhandbuch

Zielgruppe
OMNIS-MENU-Benutzer und OMNIS-MENU-Administrator

OSS (BS2000/OSD)
OSI Session Service
User Guide

Zielgruppe
OSI TP-Anwender

RSO V3.3A (BS2000/OSD)
Remote SPOOL Output
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an nichtprivilegierte Benutzer, RSO-Geräteverwalter, SPOOL-Verwalter und Systembetreuer des BS2000/OSD.

SECOS (BS2000/OSD)
Security Control System
Benutzerhandbuch

Zielgruppe
BS2000-Systemverwalter und BS2000-Anwender, die den erweiterten Zugriffsschutz für Dateien nutzen

SECOS (BS2000/OSD)
Security Control System
Tabellenheft

Zielgruppe
Erfahrene SECOS-Anwender

SM2 (BS2000/OSD)
Software Monitor
Band 1: Verwaltung und Bedienung

Zielgruppe
Anwender und Systembetreuung

TIAM (BS2000/OSD, TRANSDATA)
Benutzerhandbuch

Zielgruppe
nicht privilegierte BS2000-Anwender und Programmierer

VTSU (BS2000/OSD, TRANSDATA)
Virtual Terminal Support
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie System- und Netzverwalter

XHCS (BS2000/OSD)
8-bit-Code-Verarbeitung im BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie Systemverwalter, Anwender, die von EHCS auf XHCS umstellen

Dokumentation zum UNIX-Umfeld

CMX V6.0 (Solaris)
Betrieb und Administration
Benutzerhandbuch

Zielgruppe
Systemverwalter für Solaris

CMX V6.0 (Unix)
Betrieb und Administration
Benutzerhandbuch

Zielgruppe
Systemverwalter für UNIX-Systeme

CMX V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

Zielgruppe
Programmierer von Kommunikationsanwendungen auf UNIX-Systemen

OSS (SINIX)
OSI Session Service
User Guide

Zielgruppe
OSI TP-Anwender

PRIMECLUSTER™
Konzept (Solaris, Linux)
Benutzerhandbuch

Zielgruppe
Benutzer und Systemadministratoren

Sonstige Literatur

BS2000/OSD Technische Beschreibung
Hrsg.: Peter Jilek
Transaktionssystem
v. Rudolf Koch
ISBN 3 89578 017 0

CPI-C (X/Open)
Distributed Transaction Processing
X/Open CAE Specification, Version 2
ISBN 1 85912 135 7

Reference Model Version 2 (X/Open)
Distributed Transaction Processing
X/Open Guide
ISBN 1 85912 019 9

TX (Transaction Demarcation) (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 094 6

XATMI (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 130 6

XML
Spezifikation des W3C (www – Konsortium)
Web page: http://www.w3.org/XML

Stichwörter

32-bit-Modus 186
64-bit-Anwendung
  binden (UNIX-COBOL) 20
  ORACLE (UNIX) 75
64-bit-fähige Anwendungen
  Cobol 18
64-bit-Modus 186

A
Ablauf
  Programmaustausch 126
  Programmaustausch (Shared Objects) 141
Ablauf-Koordination, Teilprogramme 15
Ablaufverfolger
  für UTM-Anwender 163
Abmelden 81
  bei Zeitüberschreitung 111
  mit KDCOFF-Kommando 111
  von UTM-Anwendung 117
Abnormal beenden
  Anwendung 71
  Teilprogramm 148
  UTM-Datenbank-Anwendung 80
Abrechnung siehe auch Accounting
Abrechnungsphase 153, 155, 159
Abrechnungssatz 154
  Aufbau 204
Absolute Generationsnummer 123
Access-List-Konzept 109
Accounting 151
  bei verteilter Verarbeitung 161
  Festpreis 154
  Rechnerleistung 151
  verbrauchsorientiert 154
ADABAS 73
adb 64
Administrationsberechtigung 104
admlp 209
Aktivieren
  Signalbehandlung 62
Ändern
  Konto 33
Anmelden 81
  bei UTM-Anwendung 85
  fehlerhafte Eingaben 90
  Fehlversuche 90
  ohne Benutzerkennung 104
  über OSI-TP-Clients 95
  über TS-Anwendungen 93
  über UPIC-Clients 93
  über Web 96
Anmeldeverfahren
  mit SIGNON-Services 98
Anmeldeversuche
  Fehlversuche 102
  maximale Anzahl 90
Anmeldevorgang
  Anwendungsmöglichkeiten 101
  Fehlverhalten 102
  Fehlversuche 102
Anwendung
  abnormal beenden 71
  als Dienst 32
  austauschen 119
  beenden 69
  Beispielanwendung (UNIX) 210
  Beispielanwendung (Windows) 211
  mit Benutzerkennungen 108
  mit Shared Objects austauschen 145

Anwendung (Forts.)
normal beenden 70
starten mit Shared Objects 141
Anwendungsbetrieb
Vorbereitung 35
Anwendungsende
Shutdown 150
System-Absturz 150
Anwendungslogik
Teilprogramme 13
Anwendungsname
beim Anmelden 86
Anwendungsprogramm
binden (Windows) 26, 29
erzeugen 13
gebundenes ablegen (UNIX) 20
Anzahl
Asynchronvorgänge 59
Generationen 124
Prozesse bei Anwendungsstart 62
Anzahl Socket-Verbindungen
ändern 188
Anzeigen
Dateien eines Projekts 27
applifile 49
Arbeitsschritte
Anwendungsaustausch 132
ASYNTASKS 59
Aufbau
Abrechnungssatz 205
Aufbereiten
SYSLOG-Datei 42, 43
Aufbereitungstool
KDCPSYSL 43
Aufrufen
Binder (UNIX) 20
Binder (Windows) 29
Aufsteigende Generationsnummer 123
Ausgabe
letzte Dialognachricht 115
letzte wiederholen 116
Startformat 104
umlenken beim Starten (UNIX) 54
umlenken beim Starten (Windows) 56
Ausgabe (Forts.)
wiederholen 116
Auslastung
Statistik 163
Austauschen
Anwendung 119
Shared Objects 119, 139
Auswertungslisten
KDCMON 170
Automatisch starten
UTM-Anwendung 33
Automatische Größenüberwachung
SYSLOG 40
Automatisches KDCSIGN 93

B

Basis
Dateigenerationsverzeichnis 124
Basisname 35, 59
Basisnummer
Dateigenerationsverzeichnis 124
Beenden
als Dienst gestartete Anwendung 70
UTM-Anwendung 69
Beispiel
Anwendungsaustausch 132
Benutzer-Protokolldatei 45
Dateiverzeichnis PROG 124
INFORMIX-Startparameter 77
Inhalt Startparameterdatei 63
ORACLE-Startparameter 78
Shared Objects austauschen 143
SYSLOG-FGG 39
Beispielanwendung
Makefile 20
Beispielanwendung (UNIX) 210
Benutzer 152
Benutzerkennung
beim Anmelden 85
Benutzerkommandos 113
Benutzerkonto
lokal 32

Benutzer-Protokolldatei 35, 44
Beispiel 39, 45
Größe Pagepool 46
Berechtigungskonzept
Meldungen 110
Berechtigungsnachweis
automatischer 93
Berechtigungsprüfung 88, 105
Varianten 90
Betriebsmittel 152, 157
systemglobale 47
Binden
Anwendungsprogramm (UNIX) 20
Anwendungsprogramm (Windows) 29
COBOL-Programme (Windows) 31
Datenbank-Anwendung in UNIX 74
Datenbank-Anwendung in Windows 75
Dialogterminalprogramm 187
Druckerterminalprogramm 187
Makefile 20
Produktivanwendung (UNIX) 16
UTM-Datenbank-Anwendung (UNIX) 74
UTM-Datenbank-Anwendung (Windows) 28
utmwork (UNIX) 16
utmwork.exe (Windows) 29
Binder aufrufen
UNIX 20
Windows 29
Binderlauf
Installation von *open*UTM(UNIX) 187
Bindeoptionen (Windows) 27
BTRACE 59

C
C 21
C++ 21
C++-Programme
PrimePower Solaris 17
CAE-Spezifikation 73
CCmainutm 209
C-Laufzeitsystem 13
Close-String 76
cob32 18
cob64 18
COBDIR 16
COBOL-Programme
in Unix 17
in Windows 30
COBOL-Testhilfen 64
cobrtcb2 209
Console (Windows) 85
CPI-C-Beispielprogramme 209
CPICBUFFER 199
CPICDUMP 199
CPICTRACE 199

D
Darstellungsmittel 11
Dateien
notwendige für Anwendungsbetrieb 35
Dateigeneration siehe FGG
Dateigenerationsverzeichnis, siehe FGG
Dateiverzeichnis DUMP 35
Datenbank 73
Startparameter 76
Datenbank-Anwendung
binden (UNIX) 74, 75
binden (Windows) 28
Datenbank-Bibliotheken
UNIX 75
Windows 28
Datenerfassung
starten (KDCMON) 164
Datum
bei Meldungen 63
DB2 73
DB-DIAGAREA 82
dbx 64
debug 64
Debugger
starten mit (UNIX) 64
starten mit (Windows) 66
Deinstallieren
UTM-Dienst 33
DEL-Taste
ignorieren in UNIX 87

Developer Studio
Editor 25
Optionen 22
Projekt erzeugen 24
Diagnose
Diagnosedaten auf Datei schreiben 63
mit SYSLOG 35
Unterlagen bei abnormalem Beenden 72
UTM-Datenbank-Anwendung 82
UTM-Fehler 62
Dialognachrichten
letzte ausgeben 115
Dialogterminalprogramm
automatisch binden 187
Installation (UNIX) 187
Dialogterminalprozess
starten durch Benutzer 85
starten durch UNIX 87
Dienst
beenden 70
installieren 32
Name 32
starten 57
Dienstprogramm, siehe Tools
dllimport 73
Dokumentation, Wegweiser 3
Domänen-Benutzerkonto 32
Druckausgabe
ohne Drucksteuerung 208
Druckerprogramm 187
automatisch binden 187
Druckerscript 208
utmlp 208
DUMP 47
DUMP-CONTENT 60
Dump-Fehlercode 71
DUMP-MESSAGE 60
Wert zurücksetzen 60
dumpstart 209
Dynamisch
Shared Objects hinzufügen 145

E
EDITOR 198
Einrichten
FGG 127
Einschalten
Standard-Fehlerbehandlung für Signale 62
Testmodus 62
END 59
Entfernen
UTM-Dienst 33
Erste Generation 124
Erzeugen
Anwendungsprogramm 13
Meldungstexte (SYSLOG) 43
UTM-Beispielanwendung (UNIX) 210
exit()-Funktion (Windows) 28
Externverweise 120

F
Fehler
Systemumgebung 147
Fehlermeldung
beim Start 68
Fehlverhalten
des Anmeldevorgangs 102
Fehlversuche
anmelden 90
beim Anmeldevorgang 102
Festpreis-Abrechnung 154
FGG 37
Basis 124
Basis umschalten 121, 131
Benutzer-Protokolldatei 44
einrichten 127
für SYSLOG 37
Generation transferieren 129
informieren über 128
PROG 123
relativer Name 124
FILEBASE 59
filebase
DUMP 47
PROG als FGG einrichten 120
USLA 44

Funktionen
KDCPROG 127
Funktionsvarianten von UTM 114

G
gcore-Dump 62
gdb 64
Generation 123
maximale Anzahl 124
Generationsnummer
absolute 123
aufsteigende 123
relative 124
Generieren
Shared Objects 139
Gewicht 153
festlegen 157
Grace-Sign-On 90
Größe
Pagepool (Protokolldateien) 46
Größenüberwachung
automatische 40
suspendierte 40
GUID-String 79

H
Hauptprogramm 15
Hinzufügen
Programme 145

I
INFO-Datei
SYSLOG-FGG 38
INFORMIX 73, 74
Startparameter 77
install.sample 209
Installation
binden von utmdtp/utmprint bei (UNIX) 187
Installieren
UTM 185
UTM (Windows) 189
UTM-Dienst 32
INT 61
ISAM/XA 73

K
K-/F-Taste
Benutzerkommandos 113
K000 60
K001 104
K003 117
K005 110
K008 105
K009 110
K012 114
K018 117
K019 117
K020 114
K021 111
K027 103
K028 92
K043 40, 41
K046 208
K049 82
K050 68
K051 68
K068 82
K071 82
K078
gcore-Dump 62
K079 160
K080 164
K092 92
K094 90, 102
K097 89
K120 90
K121 92
K123 110
K136 36
K138 36
K155 92
Kalkulationsphase 152, 155
Kalkulationssatz 153
Aufbau 204
Kaltstart 68
KDCAPPL 119
PROG=NEW 126

KDCCSYSL
aufrufen 42
Meldungen 43
starten 42
KDCDIAG 164
KDCDISP 115
KDCEVAL
Meldungen 167
KDCFILE 35
Basisname 59
KDCLAST 116
KDCMON 163
Auswertungslisten 169
Datenerfassung starten 164
KDCOFF 117
aus Programm 111
KDCOFF BUT 117
KDCOUT 114
KDCPROG 119, 127
Beispiele 132
CREATE 127
Funktionen 127
INFO 128
SWITCH 131
TRANSFER 129
KDCPSYSL
Meldungen 43
starten 43
KDCREM 72
KDCROOT 15, 17
KDCS_C_DEBUG 195
KDCSHUT 69
KDCSIGN
automatisches 93
kdcslog 38
KDCS-Returncode 148
KDCSWTCH 104
KDCUSLOG
starten 44
Kenngrößen
KDCPROG 124
Kommentare
Startparameterdatei 58
Konfigurieren
UTM-Dienst 33
Konto
UTM-Dienst 33
KTA-Trace-Area 63
Kurzbefehl
UTM-Anwendung starten mit 56

L
LANG 84, 193
LD_LIBRARY_PATH 16
Leistungskontrolle 163
libwork.a 18
libwork.lib 27
libwork.so 18
LNK4006 29
LNK4056 29
LNK4075 29
lnkdtp.NONE 209
lnkprint.NONE 209
Lock-/Keycode-Konzept 109
Löschen
Shared Memory 72

M
Main Routine KDCROOT 15, 17
Main-Prozess
starten 53
mainutm.o 16
mainutmCC.o 16
Makefile 20
Mehrprozess-Betrieb
testen 64
Meldungen
fehlende Berechtigung 110
KDCCSYSL 43
KDCEVAL 167
KDCPSYSL 43
mit Datum und Uhrzeit 63
Meldungsmodul 13
Meldungstexte
für SYSLOG erzeugen 43
Meldungsziele
SYSLOG 35

Metasyntax 11
Microsoft SQL Server 73
Microsoft Visual C++ 21
MQSeries 73
MSGDMP 60
MSGTAC 90
MS-SQL-Server
Startparameter 79

N
Name
Dateigeneration 123
UTM-Dienst 32
Negative Abdruckquittung 208
Netzprozesse
für Socket-Verbindungen 188
nmutmwork 120
nmwork 209
Normales Beenden
Anwendung 69

O
O2 73
Objektdateien
aufnehmen in Projekt 26
Öffnen
Projekt 24
Open-String 76
*open*UTM-Anwendung, siehe UTM-Anwendung
Optionen
Developer Studio 22
ORACLE 73, 74
Startparameter 78
OSI TP
benötigte Betriebsmittel 49
OSI-TP-Clients
anmelden über 95
OSI-TP-Module, binden (UNIX) 19
OSS Lockbörse 49
OSS Shared Memory 49
OSS Taskbörse 49
OSS-Aufrufe 61
OSS-interne Trace Records 61
OSS-Tracefunktion
ein-/ausschalten 61
OTRACE 61

P
p/config 20
Passwort
beim Anmelden 88
Zeitspanne überwachen 101
PATH 55, 195, 208
PEND ER 148
Performance
Analyse über TRACE2 184
Überwachung 163
Plausibilitätsprüfung 62
Präfix
Startparameter 58
Printer-Programm
automatisch binden 187
Installation 187
Printerprozess 208
Programm
dynamisch hinzufügen 145
Programmaustausch
Ablauf 126
Beispiele 132
Shared Objects 141
UNIX 16
Programme erstellen
mit Developer Studio 25
Projekt
erzeugen mit Developer Studio 24
öffnen 24
Quellprogramm aufnehmen 26
PROT 61
Protokolldatei 44
ptermname
utmdtp 86

Q
Quick Start Kit 211

R
Readme-Dateien 7
REASON 148
Recovery Phase 68, 80
ref4utm 210
Reihenfolge
  Startparameter 58
Relative Generationsnummmer 124
Relativer FGG-Name 124
Resource Manager 73
Returncode
  Abbruch 71
RMXA 73
rootname 17, 26

S
Schreibfehler
  SYSLOG 41
sdb 64
Semaphore 47
  zurücksetzen 72
SERV 61
Service
  beenden 70
  installieren 32
  starten 57
servicename 32
SFUNC 113
Shared Memory 47
  löschen 72
Shared Objects
  Anwendung starten 141
  austauschen 119, 139, 141
  generieren 139
  unbefriedigte Externverweise 120
Shell-Umgebung 53
Shortcut erstellen
  zum Starten 56
Shortcuts 190
Signalbehandlung 148
  Standardfehlerbehandlung ausschalten 62
SIGNON-Service
  bei UTM-Datenbank-Anwendung 81
SIGNON-Services 98
Socket-Netzprozess 188
  austauschen 188
Speicherverbrauch
  SYSLOG-FGG 41
SPI 61
Sprache
  UTM-Meldungen 84
SQL-Server 73
Standard-Fehlerbehandlung
  Signale 62
START 59
Startart
  UTM-Dienst 33
Starten
  KDCCSYSL 42
  KDCPSYSL 43
  KDCUSLOG 44
  Mehrprozess-Anwendung 64, 66
  UTM-Anwendung 53
  UTM-Anwendung als Dienst 57
  UTM-Anwendung in UNIX 53
  UTM-Anwendung in Windows 55
  UTM-Anwendung mit Shared Objects 141
  UTM-Anwendung über Shortcut 56
Startfehlercode 58 37
Startformat
  ausgeben 104
START-Kommando 59
Startkommandos 59
Startparameter
  INFORMIX 77
  ORACLE 78
  Präfix 58
  Syntax 58
  TEST 64, 66
  UTM 58
  UTM-Datenbank-Anwendung 76
Startparameterdatei
  UTM-Anwendung 58
stat2dyn 210
Statistik
  Anmeldeversuche 101
  Auslastung 163

Statusinformationen
applifile 49
stderr
UNIX 54
Windows 55
stdout
UNIX 54
Windows 55
SYBASE 73
Symboltabelle 20, 120
SYSLOG
als einfache Datei 36
als FGG 37
Größenüberwachung 41
Schreibfehler 41
umschalten 40, 41
Verhalten bei Schreibfehlern 41
SYSLOG-Datei 35
aufbereiten 42, 43
SYSLOG-FGG
anlegen 38
automatische Größenüberwachung 40
Beispiel 39
Speicherverbrauch 41
suspendierte Größenüberwachung 40
Systemfehler 72
Systemglobale Betriebsmittel 47
Systemkonto 32
System-Protokolldatei 35
SYSLOG 35
Szenarien
UTM-Berechtigungsprüfung 90

T

Task
starten 62
TASK PERFORMANCE TRACE 184
TASKS-IN-PGWT 62
Tastaturbefehl
starten Anwendung mit (Windows) 56
Teilprogramm 13
abnormal beenden 148
binden (UNIX) 17
Ende erfassen (KDCMON) 184
Teilprogramm (Forts.)
in Projekt aufnehmen (Windows) 26
Start erfassen (KDCMON) 184
TEMP 203
Terminal-Benutzer
anmelden an *open*UTM 85
Terminals
anmelden an *open*UTM über 84
TEST 64, 66
Testen
Mehrprozess-Betrieb 64
UTM-Produktivanwendung (UNIX) 64
UTM-Produktivanwendung (Windows) 66
Workprozess neu starten 65
Testhilfen
in UNIX 64
TESTMODE 62
Testmodus 62
Diagnosedaten auf Datei schreiben 63
einschalten 62
TMP 203
tns4utm 210
Tool
KDCPROG 127
KDCPSYSL 43
KDCREM 72
KDCSHUT 69
TRACE2 184
Tracefunktion BCAM ein-/ausschalten 59
Tracefunktion OSS
ein-/ausschalten 61
Trace-Informationen
schreiben 63
Trace-Records 61
TS-Anwendungen
anmelden über 93
TXPATH 200
TXTRACE 199

U

U02 110
U16 110
U244 64, 66

Übersetzen
COBOL-Programme 30
ROOT-Tabellen-Source 13
Überwachung
Performance 163
Uhrzeit
bei Meldungen 63
Umgebungsvariable 192
CPIC 198
Dienstprogramm kdcdump 198
Netzprozesse 197
*open*UTM 192
TX 199
utmwork-Prozesse 195
Windows 202
XATMI 200
Umschalten
SYSLOG-Datei 36, 40
Unterbrochener Vorgang 104
UPIC-Clients
anmelden 93
USERNAME 202
UTM
installieren (UNIX) 185
installieren (Windows) 189
Systemfunktionen 15
UTM_ABORT_WITH_EXCEPTION 195
UTM_APPLI_CONLET 197
UTM_BREAK_BEFORE_DUMP 203
UTM_BREAK_BEFORE_KCSTRMA 203
UTM_CORE_DUMP 194
UTM_IPC_EXTP_LETTER 194
UTM_IPC_LETTER 193
UTM_MAIN_KILL_TIME 202
UTM_MSG_DATE 194
UTM_NET_COMMON_WAIT 197
UTM_NET_SELECT_TIME 197
UTM_OSS_SHM_BASE 196
UTM_PIPE_TIME 202
UTM-Anwendung
abnormal beenden 71
beenden 69
starten 53
Warmstart 68
UTM-Beispielanwendung
UNIX 210
Windows 211
UTM-Benutzerkommandos 113
UTM-Berechtigungsprüfung 105
UTM-Bibliothek
UNIX 18
Windows 27
utm-dateiverzeichnis
UNIX 186
Windows 189
utm-dateiverzeichnis/shsc 209
UTM-Datenbank-Anwendung 73
abnormal beenden 80
Diagnose 82
UTM-Datenbank-Anwendung binden
UNIX 74
Windows 75
UTM-Dienst 32
beenden 70
deinstallieren 33
installieren 32
konfigurieren 33
starten 57
utmdtp 85
automatisch binden 187
starten durch UNIX 87
UTM-Dump 71, 147
bei inkompatiblem KDCS-Returncode 61
bei K-Meldung 60
bei kompatiblem KDCS-Returncode 60
bei SIGN-Aufruf 61
UTM-F-Anwendung
asynchrone Nachrichten 114
UTM-Funktionsaufrufe
erfassen 184
UTM-Installationsverzeichnis
UNIX 186
Wndows 189
utmlp 208, 210
Druckershellscript 208
utmmain 53
Filter für Ausgabe (UNIX) 54, 56
starten in UNIX 53

utmmain (Forts.)
starten in Windows 55
utmmains 32
UTM-Meldungen
Sprache 84
UTM-Messmonitor 163
UTM-MSG-DATE 63
UTM-Objektdateien
UNIX 16
Windows 26
UTMPATH 53, 186, 192
utmprint
automatisch binden 187
UTM-Produktivanwendung
testen (UNIX) 64
testen (Windows) 66
utmsample 210
UTM-S-Anwendung
ansynchrone Nachrichten 114
UTM-Systemmodule
UNIX 18
Windows 27
UTMTRAC 193
utmwork 20
binden (UNIX) 16
binden (Windows) 29
in FGG übertragen 129
utmwork.exe 29

V
Varianten
Berechtigungsprüfung 90
Verbindungs-Benutzerkennung 93
Verbrauchsorientiertes Accounting 154
Verletzung der Zugriffsrechte
Meldungen 110
Verrechnungseinheiten 154
Zähler 154
Versionskonzept
Shared Objects 139
Verteilte Verarbeitung
Accounting 161
Vorgang
Benutzerkennung 108
Vorgänge starten
vom OSI-TP-Client aus 107
von TS-Anwendungen aus 107
von UPIC aus 107
Vorgangswiederanlauf 108

W
Warmstart 68, 72
UTM-Datenbank-Anwendung 80
Web
anmelden über 96
Web*Transactions* 96
Wechselpufferverarbeitung
SYSLOG 35
Wiederanlauf 68, 104
Wiederholen
Ausgabe 116
Workprozesse
Anzahl 62
beenden ohne Signalbehandlung 62

X
xa_switch_t-Struktur 73
XAPTP Shared Memory 49
XAP-TP System Programming Interface 61
XAP-TP-Baustein 61, 62
XA-Schnittstelle 73
XATMI-Bibliothek 28
xdb 64
XTLCF 200
XTPALCF 200
XTPATH 201
XTSVRTR 201

Z
Zeitüberwachung 111
Zugangskontrolle 88
Zurücksetzen
DUMP-MESSAGE-Wert 60
Zwischendialog
beim Anmelden 99

Inhalt

1 Einleitung 1
1.1 Konzept und Zielgruppen dieses Handbuchs 2
1.2 Wegweiser durch die Dokumentation zu *open*UTM 3
1.2.1 *open*UTM-Dokumentation 3
1.2.2 Dokumentation zum openSEAS-Produktumfeld 6
1.2.3 Readme-Dateien 7
1.3 Neuerungen in *open*UTM V5.2 8
1.3.1 Neue Plattformen 8
1.3.2 Neue Server-Funktionen 8
1.3.3 Neue Client-Funktionen 10
1.3.4 Erweiterungen für *open*UTM WinAdmin 10
1.4 Darstellungsmittel 11

2 Anwendungsprogramm erzeugen 13
2.1 UTM-Prozess binden unter UNIX 16
2.1.1 Benötigte Objektmodule 16
2.1.2 Binder aufrufen 20
2.1.3 Tätigkeiten nach dem Binden 20
2.1.4 Binden mit Makefile 20
2.2 Anwendungsprogramm erzeugen unter Windows 21
2.2.1 Optionen des Developer Studios einstellen 22
2.2.2 Projekt erzeugen 24
2.2.3 Quellprogramme schreiben 25
2.2.4 Anwendung übersetzen und binden 26
2.2.5 Anwendungsprogramme als DLLs erstellen 30
2.2.6 COBOL-Anwendungsprogramme in Windows 30
2.2.7 Anwendung als Dienst installieren 32

3 Notwendige Dateien und systemglobale Betriebsmittel 35
3.1 System-Protokolldatei SYSLOG 35
3.1.1 SYSLOG als einfache Datei 36
3.1.2 SYSLOG als Dateigenerationsverzeichnis 37
3.1.3 Das Tool KDCSLOG zum Anlegen der SYSLOG-FGG 38
3.1.3.1 Automatische Größenüberwachung 40
3.1.4 Schutz vor zu großer SYSLOG-Datei 41
3.1.5 Verhalten bei Schreibfehlern 41

3.1.6 SYSLOG-Datei auswerten . . . 42
3.1.6.1 Das Tool KDCCSYSL - SYSLOG-Datei aufbereiten . . . 42
3.1.6.2 Das Tool KDCPSYSL - Meldungstexte erzeugen . . . 43
3.1.6.3 Meldungen von KDCCSYSL und KDCPSYSL . . . 43
3.2 Benutzer-Protokolldatei . . . 44
3.2.1 Verhalten bei Schreibfehlern . . . 46
3.3 Dateiverzeichnis DUMP . . . 47
3.4 Systemglobale Betriebsmittel einer Anwendung . . . 47
3.4.1 Von einer UTM-Anwendung benötigte Betriebsmittel . . . 47
3.4.2 Performance-Verbesserung: Größe der Datenbereiche im IPC-Shared-Memory anpassen . . . 50
3.4.2.1 Maximalen Datenbereich für das Nachrichtenaufkommen pro Verbindung anpassen . . . 51

4 UTM-Anwendung starten . . . 53
4.1 Starten einer UTM-Anwendung in UNIX . . . 53
4.2 Starten einer UTM-Anwendung in Windows . . . 55
4.2.1 Starten mit utmmain . . . 55
4.2.2 Starten als Dienst . . . 57
4.3 Startparameterdatei der Anwendung . . . 58
4.3.1 Startparameter für *open*UTM . . . 58
4.4 Starten einer UTM-Anwendung mit Debugger . . . 64
4.4.1 Starten mit Debugger in UNIX . . . 64
4.4.2 Starten mit Debugger in Windows . . . 66
4.5 Kaltstart und Warmstart . . . 68
4.6 Fehlermeldungen beim Start . . . 68

5 UTM-Anwendung beenden . . . 69
5.1 UTM-Anwendung normal beenden . . . 69
5.2 Das Tool KDCSHUT - UTM-Anwendung auf Shell-Ebene normal beenden . . . 70
5.3 Dienst beenden in Windows . . . 70
5.4 UTM-Anwendung abnormal beenden . . . 71
5.5 Das Tool KDCREM . . . 72

6 UTM-Datenbank-Anwendung . . . 73
6.1 UTM-Datenbank-Anschluss generieren . . . 73
6.2 UTM-Datenbank-Anwendung binden in UNIX . . . 74
6.3 UTM-Datenbank-Anwendung binden in Windows . . . 75
6.4 UTM-Datenbank-Anwendung starten und beenden . . . 76
6.4.1 Startparameter für eine UTM-Datenbank-Anwendung . . . 76
6.4.2 UTM-Datenbank-Anwendung normal beenden . . . 80
6.4.3 UTM-Datenbank-Anwendung abnormal beenden . . . 80
6.5 Betrieb einer UTM-Datenbank-Anwendung . . . 81
6.5.1 Anmelden und Abmelden eine Benutzers . . . 81
6.5.2 Diagnose . . . 82

7 Arbeiten mit einer UTM-Anwendung 83
7.1 Anmeldeverfahren mit Benutzerkennungen 84
7.1.1 Standard-Anmeldeverfahren für Terminals 84
7.1.1.1 Starten des Dialogterminalprozesses durch den Benutzer 85
7.1.1.2 Starten des Dialogterminalprozesses durch UNIX 87
7.1.1.3 Standard-Anmeldedialog 88
7.1.1.4 Automatisches KDCSIGN 93
7.1.2 Anmeldeverfahren für UPIC-Clients und TS-Anwendungen 93
7.1.3 Anmeldeverfahren für OSI-TP-Clients 95
7.1.4 Anmeldeverfahren im World Wide Web über Web*Transactions* 96
7.1.5 Mehrfach-Anmeldungen unter einer Benutzerkennung 97
7.1.6 Anmeldeverfahren mit Anmeldevorgängen 98
7.1.6.1 Anmeldevorgang für Terminals 99
7.1.6.2 Anmeldevorgang für TS-Anwendungen 99
7.1.6.3 Anmeldevorgang für UPIC-Clients 100
7.1.6.4 Anwendungsmöglichkeiten für Anmeldevorgänge 101
7.1.6.5 Eigenschaften von Anmeldevorgängen 101
7.1.7 Verhalten bei gesperrten Clients/LTERM-Partnern 103
7.2 Anmeldeverfahren ohne Benutzerkennungen 104
7.3 UTM-Services aufrufen 105
7.3.1 Vorgänge vom Terminal aus starten 105
7.3.2 Vorgänge vom UPIC-Client und OSI-TP-Client aus starten 107
7.3.3 Vorgänge von TS-Anwendungen aus starten 107
7.3.4 Vorgangswiederanlauf 108
7.4 Berechtigungskonzept von *open*UTM 109
7.5 Abmelden von der UTM-Anwendung 111
7.6 UTM-Benutzerkommandos für Terminals 113
KDCOUT - Asynchrone Nachricht ausgeben 114
KDCDISP - Letzte Dialognachricht ausgeben 115
KDCLAST - Letzte Ausgabe wiederholen 116
KDCOFF - Abmelden von einer UTM-Anwendung 117

8 Programmaustausch im Betrieb 119
8.1 Anwendung austauschen 119
8.1.1 Voraussetzungen für den Anwendungsaustausch 120
8.1.2 Dateigenerationsverzeichnis PROG 123
8.1.3 Ablauf des Anwendungsaustausches 126
8.1.4 Das Tool KDCPROG 127
CREATE - Dateigenerationsverzeichnis (FGG) einrichten 127
INFO - Aktuellen Zustand der FGG abfragen 128
TRANSFER - utmwork in die FGG übertragen 129
SWITCH - Basis der FGG umschalten 131
8.1.5 Beispiel für Anwendungsaustausch 132

8.2 Shared Objects austauschen . . . 139
8.2.1 Shared Objects bereitstellen und generieren . . . 139
8.2.2 Start der Anwendung . . . 141
8.2.3 Ablauf des Austausches . . . 141
8.2.3.1 Shared Objects mit LOAD-MODE=STARTUP austauschen . . . 141
8.2.3.2 Shared Objects mit LOAD-MODE=ONCALL austauschen . . . 142
8.2.4 Beispiele für Austausch von Shared Objects . . . 143
8.2.5 Anwendung mit Shared Objects austauschen . . . 145
8.2.6 Programme dynamisch hinzufügen . . . 145

9 Fehlertoleranz von *open*UTM . . . 147
9.1 Fehler, die *open*UTM erkennt . . . 148
9.2 Reaktion von *open*UTM auf Signale . . . 148
9.3 Anwendungsende durch System-Absturz/Shutdown . . . 150

10 Accounting . . . 151
10.1 Begriffsdefinitionen . . . 152
10.2 Phasen des Accounting . . . 155
10.2.1 Kalkulationsphase . . . 155
10.2.2 Variante des Abrechnungsverfahrens festlegen . . . 156
10.2.3 Abrechnungsphase . . . 158
10.2.4 Auswertung . . . 160
10.2.5 Fehlersituationen . . . 160
10.3 Abrechnung bei verteilter Verarbeitung . . . 161
10.4 Einschränkungen . . . 162

11 Leistungskontrolle mit KDCMON . . . 163
11.1 Erfassung starten und beenden . . . 164
11.2 Daten auswerten mit KDCEVAL . . . 165
11.3 Auswertungsdaten auf dem PC bearbeiten . . . 168
11.4 Auswertungslisten . . . 169
TASKS: UTILIZATION OF THE UTM TASKS . . . 171
SUMM: TRANSACTION EVALUATION . . . 172
TIMES: DISTRIBUTION OF PROCESSING TIMES . . . 173
KCOP: UTM CALLS STATISTIC . . . 174
WAIT: WAITING TIMES . . . 176
TCLASS: EVALUATION OF THE TAC CLASSES . . . 177
TACCL: TAC SPECIFIC TAC CLASS EVALUATION . . . 179
TACPT: TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES . . . 180
TACLIST: TAC SPECIFIC STATISTICS . . . 181
TRACE: TASK SPECIFIC TRACES . . . 182
TRACE2: TASK PERFORMANCE TRACE . . . 184

12 Anhang . . . 185
12.1 *open*UTM installieren in UNIX . . . 185
12.1.1 UTM-Systemfunktionen in UNIX installieren . . . 186
12.1.2 Dialogterminal- und Printer-Programm binden . . . 187
12.1.2.1 Standard-Binderlauf bei Installation . . . 187
12.1.2.2 Programme ohne Formatierung nachträglich binden . . . 187
12.1.3 Unterschiedliche Socket-Netzprozesse einsetzen . . . 188
12.2 *open*UTM installieren unter Windows . . . 189
12.2.1 Installation des *open*UTM-Servers V5.2 . . . 189
12.2.2 Benutzerumgebung . . . 190
12.3 Struktur des UTM-Installationsverzeichnisses . . . 191
12.4 Umgebungsvariablen einer UTM-Anwendung . . . 192
12.4.1 Allgemeine Umgebungsvariablen für *open*UTM unter UNIX und Windows . . . 192
12.4.2 Umgebungsvariablen für Workprozesse . . . 195
12.4.3 Umgebungsvariablen für Netzprozesse . . . 197
12.4.4 Umgebungsvariable für das Tool KDCDUMP . . . 198
12.4.5 Umgebungsvariablen für die X/Open Schnittstellen . . . 198
12.4.5.1 CPIC . . . 198
12.4.5.2 TX . . . 199
12.4.5.3 XATMI . . . 200
12.4.6 Zusätzliche Umgebungsvariablen für *open*UTM unter Windows . . . 202
12.5 Aufbau der Accountingsätze von *open*UTM . . . 204
12.5.1 Aufbau des Abrechnungssatzes . . . 205
12.5.2 Aufbau des Kalkulationssatzes . . . 206
12.6 Druckausgaben ohne Druckersteuerung abwickeln (UNIX) . . . 208
12.7 Beispielprogramme und Beispielanwendungen . . . 209
12.7.1 CPI-C-Beispielprogramme . . . 209
12.7.2 Beispielprozeduren in UNIX . . . 209
12.7.3 *open*UTM(UNIX)-Beispielanwendung . . . 210
12.7.4 *open*UTM(Windows) - Quick Start Kit . . . 211

Fachwörter . . . 213

Abkürzungen . . . 247

Literatur . . . 253

Stichwörter . . . 265

*open*UTM V5.2 (UNIX, Windows)
Einsatz von *open*UTM-Anwendungen unter UNIX und Windows

Benutzerhandbuch

Zielgruppe
Dieses Handbuch richtet sich an Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen.

Inhalt
Es enthält alle Informationen, um ein UTM-Anwendungsprogramm in UNIX und Windows zu erzeugen und eine UTM-Anwendung einzusetzen. Die Besonderheiten, die Sie beim Betrieb einer UTM-Datenbank-Anwendung beachten müssen, sind zentral in einem gleichlautenden Kapitel zusammengestellt. Ausführlich wird darauf eingegangen, wie sich Terminal-Benutzer und andere Clients an eine UTM-Anwendung anmelden können.
Zusätzlich gibt es eigene Kapitel über die Tools, die Ihnen für den Betrieb und die Kontrolle einer UTM-Produktivanwendung zur Verfügung stehen.

Ausgabe: April 2004

Datei: utmx_betr.pdf

Copyright © Fujitsu Siemens Computers GmbH, 2004.

Alle Rechte vorbehalten.
Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.

Dieses Handbuch wurde erstellt von
cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Fujitsu Siemens Computers GmbH
Handbuchredaktion
81730 München

Kritik
Anregungen
Korrekturen

Fax: 0 700 / 372 00000

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Absender

Kommentar zu *open*UTM V5.2
Einsatz von *open*UTM-Anwendungen unter UNIX und Windows

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009